10 PROGRAM ENVIRONMENT AND INITIALIZATION

This section discusses possible alternative software environments using OS Configurations. Environments other than those discussed here are also possible. A thorough understanding of the power-up and system reset processes (see Section 7) will be necessary to evaluate all alternative environments.

CARTRIDGE

Most games (and some language processors) are supported via the cartridge environment. The cartridge resident software is in control of the system, sometimes using the OS and sometimes not. A cartridge can specify whether the diskette is to be booted at power-up time, whether the cartridge is to provide the controlling software, or whether the cartridge is a special diagnostic cartridge. These options are specified by bits in the cartridge header, as shown below:

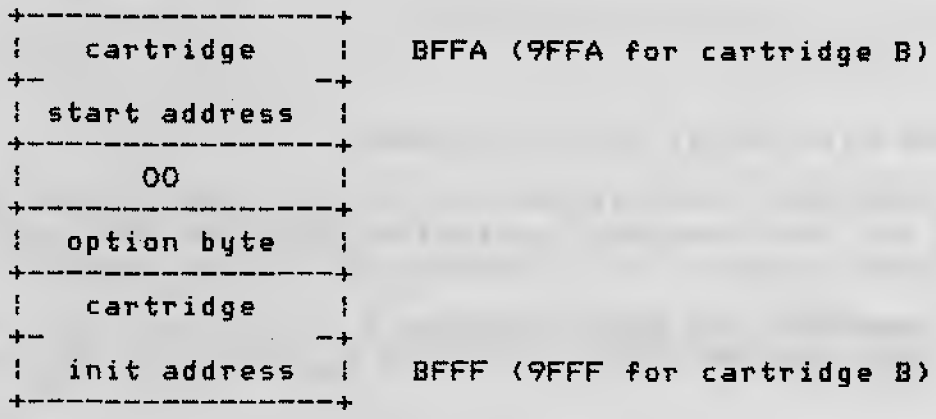


Figure 10-1 Cartridge Header Format

The byte of "00" is used to allow the OS to determine when a cartridge is inserted; locations BFFC and 9FFC will not read zero when there is neither RAM at those locations nor a cartridge inserted. RAM is differentiated from a cartridge by its ability to be altered.

The option byte has the following option bits:

bit 0 = 0, then do not boot the diskette.
1, then boot the diskette.

Bit 2 = 0, then init but do not start the cartridge.
1, then init and start the cartridge.

bit 7 = 0, then cartridge is not a diagnostic cartridge.
1, then cartridge is a diagnostic cartridge and control will be given to the cartridge before any of the OS is initialized (JMP (BFFE)).

The cartridge init address specifies the location to which the OS will JSR during all power-up and system reset operations. As a minimum, this vector should point to an RTS instruction.

The cartridge start address specifies the location to which the OS will JMP during all power-up and system reset operations, if bit 1 of the option byte is = 1. The application should examine the variable WARMST [0008] if system reset action is to be different than power-up (WARMST will be zero on power-up and nonzero thereafter).

Cartridge Without Booted Support Package

A cartridge that does not specify the diskette-boot option and does not support the cassette-boot possibility can use lower memory (from 0480 to the address in MEMTOP [02E5]) in any way it sees fit.

Cartridge With Booted Support Package

A cartridge that does specify the diskette-boot option or does support the cassette-boot possibility must use some care in its use of lower memory. The following regions are defined:

0480-06FF is always available to the cartridge.
MEMLO/MEMTOP region is always available to the cartridge.

DISKETTE-BOOTED SOFTWARE

Software can be booted from the disk drive at power-up time in response to one of the following conditions:

Neither Cartridge A nor B is inserted.

Cartridge A is inserted and has bit 0 of its option byte [BFFD] = 1.

Cartridge B is inserted and has bit 0 of its option byte [9FFD] = 1.

If any of these conditions are met, the OS will attempt to read the boot record from sector #1 of disk drive 1 and then transfer control to the software that was read in. The exact sequence of operations will be explained later in this section.

Diskette-Boot File Format

The key region of a diskette-boot file is the first six bytes, which are formatted as shown below:

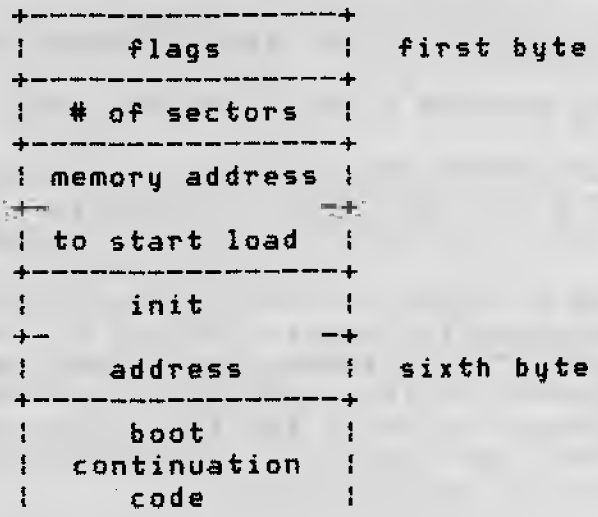


Figure 10-2 Diskette-Boot File Format

The first byte is stored in DFLAGS [0240], but is otherwise unused. It should equal zero.

The second byte contains the number of 128-byte diskette sectors to be read as part of the boot process (including the record containing this information). This number can range from 1 to 255, with 0 meaning 256.

The third and fourth bytes contain the address (lo,hi) at which to start loading the first byte of the file.

The fifth and sixth bytes contain the address (lo,hi) to which the booter will transfer control after the boot process is complete and whenever the [SYSTEM.RESET] key is pressed.

The Diskette File Management System (FMS) has extra bytes assigned to its boot record, but this is a special case of the generalized diskette-boot and is discussed in Section 5.

Diskette-Boot Process

If no cartridge is installed, then the diskette will follow these steps to boot up:

1. Read the first diskette record to the cassette buffer [0400].

2. Extract information from the first six bytes:

   Save the flags byte to DFLAGS [0240,1]. Save the # of sectors to boot to DBSECT [0241,1]. Save the load address to BOOTAD [0242,2]. Save the initialization address in DOSINI [000C,2].

3. Move the record just read to the load address specified.

4. Read the remaining records directly to the load area.

5. JSR to the load address+6 where a multistage boot process can continue. The carry bit indicates the success of this operation (carry set = error, carry reset = success).

   NOTE: During step 5, after the initial boot process is complete, the booter will transfer control to the seventh byte of the first record. The software should continue the boot process at this point, if it is a multistage boot. The value of MEMLO [02E7] should point to the first free RAM location beyond the software just booted. It should be established by the booted software as shown below:

```
LDA     #END+1          ; SET UP LSB.
STA     MEMLO
STA     APPMHI
LDA     #END+1/256      ; SET UP MSB.
STA     MEMLO+1
STA     APPMHI+1
```

   If the booted software is to take control of the system at the end of the boot operation, the vector DOSVEC [000A] must be set up by the application at this time; DOSVEC points to the

restart entry for the booted application. If the booted software is not to take control, then DOSVEC should remain unchanged.

```
LDA     #RESTRT          ; RESTART LSB.
STA     DOSVEC
LDA     #RESTRT/256
STA     DOSVEC+1
```

6. JSR indirectly through DOSINI for initialization of the application; the application will initialize and return.

   NOTE: The OS enters the initialization point on every system reset and power-up. Internal initialization can take place during system reset and power-up as well. Initialization can also be deferred until Step 7 for controlling applications.

7. JMP indirectly through DOSVEC to transfer control to the application.

   NOTE: Pressing the [SYSTEM.RESET] key after the application is fully booted will cause steps 6 and 7 to be repeated.

Sample Diskette-Bootable Program Listing

This skeletal program can be booted from the diskette. It retains control when it is entered.

```
; THIS IS THE START OF THE PROGRAM FILE.

PST=    $0700                   ; (OR SOME OTHER LOCATION).
*=      PST                     ; (.ORG).

; THIS IS THE diskette-boot CONTROL INFORMATION.

        .BYTE   0               ;
        .BYTE   PND-PST+127/128 ; NUMBER OF RECORDS.
        .WORD   PST             ; MEMORY ADDRESS TO START LOAD.
        .WORD   PINIT           ; PROGRAM INIT.
```

```
; THIS IS THE START OF THE BOOT CONTINUATION.

        LDA     #PND            ; ESTABLISH LOW MEMORY LIMITS.
        STA     MEMLO
        STA     APPMHI
        LDA     #PND/256
        STA     MEMLO+1
        STA     APPMHI+1

        LDA     #RESTRT         ; ESTABLISH RESTART VECTOR.
        STA     DOSVEC
        LDA     #RESTRT/256
        STA     DOSVEC+1

        CLC                     ; SET FLAG FOR SUCCESSFUL BOOT.
        RTS

; APPLICATION INITIALIZATION ENTRY POINT.

PINIT   RTS                     ; NOTHING TO DO HERE FOR ...
                                ; ... CONTROLLING APPLICATION.

; THE MAIN BODY OF THE PROGRAM FOLLOWS.

RESTRT=*

; THE MAIN BODY OF THE PROGRAM ENDS HERE.

PND=    *                       ; 'PND' = NEXT FREE LOCATION.
        .END
```

Figure 10-3 Diskette-Bootable Program Listing Example

Program to Create Diskette-Boot Files

This section provides a program that can be used to make bootable files on diskettes. The program given is not the only one possible, and no claims are made as to its elegance.

Shown below is a listing of the program to create diskette-boot files.

```
; THIS PROGRAM WRITES A SINGLE "FILE" TO THE DISKETTE AND IS
; USED IN CONJUNCTION WITH A PROCEDURE TO MAKE DISKETTE-
; BOOTABLE FILES.  THE FOLLOWING TWO SYMBOLS MUST BE EQUATED
; USING THE MEMORY LIMITS OF THE PROGRAM TO BE COPIED:
;
;      'PST' = PROGRAM START ADDRESS (SEE SAMPLE PROGRAM).
;      'PND' = PROGRAM END ADDRESS (SEE SAMPLE PROGRAM).


SECSIZ=128                          ; DISKETTE SECTOR SIZE.
PST=    $0700
PND=    $1324
FLEN=   PND-PST+SECSIZ-1/SECSIZ ; # OF SECTORS IN FILE.

*=      $B000                       ; THIS PROGRAM'S ORIGIN.



BOOTB   BRK                         ; *** LOAD APPLICATION ***

; SET UP DEVICE CONTROL BLOCK FOR DISKETTE HANDLER CALL

        LDA     #FLEN               ; # OF SECTORS TO WRITE.
        STA     COUNT

        LDA     #1                  ; DISK DRIVE #1.
        STA     DUNIT

        LDA     #'W                 ; SET UP FOR WRITE WITH CHECK.
        STA     DCOMND

        LDA     #PST                ; POINT TO START OF APPLIC. PROG.
        STA     DBUFLO
        LDA     #PST/256
        STA     DBUFHI

        LDA     #01                 ; SET UP STARTING SECTOR # = 0001.
        STA     DAUX1
        LDA     #00
        STA     DAUX2
```

```
; NOW WRITE THE FILE ONE SECTOR AT A TIME.

BOT010  JSR     DSKINV          ; WRITE ONE SECTOR.
        BMI     DERR            ; ERROR.

        LDA     DBUFLO          ; INCREMENT MEMORY ADDRESS.
        CLC
        ADC     #SECSIZ
        STA     DBUFLO
        LDA     DBUFHI
        ADC     #0
        STA     DBUFHI

        INC     DAUX1           ; INCREMENT SECTOR #.
        BNE     BOT020
        INC     DAUX2

BOT020  DEC     COUNT           ; MORE SECTORS TO WRITE?
        BNE     BOT010          ; YES.

        BRK                     ; STOP WHEN DONE.

DERR    BRK                     ; STOP ON ERROR.

COUNT   *=*+1                   ; SECTOR COUNT.

; THIS IS THE CARTRIDGE HEADER

*=      $BFF9                   ; "A" CARTRIDGE.

INIT    RTS
        .WORD   BOOTB
        .BYTE   0,4
        .WORD   INIT

        .END
```

CASSETTE-BOOTED SOFTWARE

You can boot software from the cassette as well as from the diskette, at power-up. The following requirements must be met in order to boot from the cassette:

- You must be pressing the [START] key as power is applied to the system.

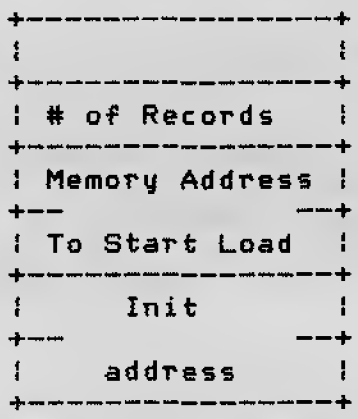
- A cassette tape with a proper boot format file must be installed in the cassette drive, and the PLAY button must be pressed.

- When you are given the audio prompt by the cassette handler you must press the [RETURN] key.

If all of these conditions are met, the OS will read the boot file from the cassette and then transfer control to the software that was read in. The exact sequence of operations will be explained later in this section.

Cassette-Boot File Format

The key region of a cassette-boot file is the first six bytes, that are formatted as shown below:



The first byte is not used by the cassette-boot process.

The second byte contains the number of 128-byte cassette records to be read as part of the boot process (including the record containing this information). This number can range from 1 to 255, with 0 meaning 256.

The third and fourth bytes contain the address (lo,hi) to which the booter will transfer control after the boot process is complete and whenever the [SYSTEM.RESET] key is pressed.

Cassette-Boot Process

The cassette-boot process is described step-by-step for a configuration in that no cartridge is installed and no diskettes are attached. For the general case see Section 7.

1. Read the first cassette record to the cassette buffer.

2. Extract information from the first six bytes:

Save the # of records to boot. Save the load address. Save the initialization address in CASINI [0002]

3. Move the record just read to the load address specified.

4. Read the remaining records directly to the load area.

5. JSR to the load address+6 where a multistage boot process can continue; the carry bit will indicate the success of this operation (carry set=error, carry reset=success).

6. JSR indirectly through CASINI for initialization of the application; the application will initialize and return.

7. JMP indirectly through DOSVEC to transfer control to the application.

Pressing the [SYSTEM.RESET] key after the application is fully booted will cause steps 6 and 7 to be repeated.

NOTE: After the initial boot process is complete, the booter will transfer control to the seventh byte of the first record; at this point the software should continue the boot process (if it is a multistage boot) and then stop the cassette drive, which due to a system bug will still be running, using the following instruction sequence:

```
        LDA     #$3C
        STA     PACTL [D302]
```

The application should then set a value in MEMLO [0237] that points to the first free RAM location beyond the software just booted, as shown below:

```
        LDA     #END+1
        STA     MEMLO
        STA     APPMHI
        LDA     #END+1/256
        STA     MEMLO+1
        STA     APPMHI+1
```

If the booted software is to take control of the system at the end of the boot operation, the vector DOSVEC [000A] must be set up by the application at this time; DOSVEC points to the restart entry for the booted application. If the booted software is not to take control, then DOSVEC should remain unchanged.

```
   LDA    #RESTRT              ; RESTART LSB
   STA    DOSVEC
   LDA    #RESTRT/256
   STA    DOSVEC+1
```

NOTE: The initialization point is entered on every system reset and power-up; internal initialization can take place here.

For controlling applications initialization can also be deferred until step 7.

Sample Cassette-Bootable Program Listing

Shown below is a skeletal program that can be booted from the cassette and that retains control when it is entered.

```
; THIS IS THE START OF THE PROGRAM FILE.

PST=    $0700                   ; (OR SOME OTHER LOCATION).
*=      PST                     ; (.ORG).


; THIS IS THE cassette-boot CONTROL INFORMATION.

        .BYTE   0               ; (DOESN'T MATTER).
        .BYTE   PND-PST+127/128 ; NUMBER OF RECORDS.
        .WORD   PST             ; MEMORY ADDRESS TO START LOAD.
        .WORD   PINIT           ; PROGRAM INIT.

; THIS IS THE START OF THE BOOT CONTINUATION.

        LDA     #$3C            ; STOP THE CASSETTE.
        STA     PACTL

        LDA     #PND            ; ESTABLISH LOW MEMORY LIMITS.
        STA     MEMLO
        STA     APPMHI
        LDA     #PND/256
        STA     MEMLO+1
        STA     APPMHI+1

        LDA     #RESTRT         ; ESTABLISH RESTART VECTOR.
        STA     DOSVEC
        LDA     #RESTRT/256
        STA     DOSVEC+1

        CLC                     ; SET FLAG FOR SUCCESSFUL BOOT.
        RTS

; APPLICATION INITIALIZATION ENTRY POINT.

PINIT   RTS                     ; NOTHING TO DO HERE FOR ...
                                ; ... CONTROLLING APPLICATION.

; THE MAIN BODY OF THE PROGRAM FOLLOWS.

RESTRT=*

; THE MAIN BODY OF THE PROGRAM ENDS HERE.
```

```
PND=    *                       ; 'PND' = NEXT FREE LOCATION.
        .END
```

Figure 10-4 Sample Cassette-Bootable Program

Program to Create Cassette-Boot Files

This section provides a program listing that can be used to make bootable files on cassette tapes. The program given is not the only one possible, and no claims are made as to its elegance.

Shown below is a listing of the program to create a cassette-boot file:

```
;   THIS PROGRAM WRITES A SINGLE FILE TO THE CASSETTE AND IS
;   USED IN CONJUNCTION WITH A PROCEDURE TO MAKE CASSETTE-
;   BOOTABLE FILES.  THE FOLLOWING TWO SYMBOLS MUST BE EQUATED
;   USING THE MEMORY LIMITS OF THE PROGRAM TO BE COPIED:

;       'PST' = PROGRAM START ADDRESS (SEE SAMPLE PROGRAM).
;       'PND' = PROGRAM END ADDRESS (SEE SAMPLE PROGRAM).

PST=    $0700
PND=    $1324
FLEN=   PND-PST+127/128*128     ; ROUND UP TO MULTIPLE OF 128.

*=      $8000                   ; THIS PROGRAM'S ORIGIN.

BOOTB   LDX     #$10            ; USE IOCB #1.

; FIRST OPEN THE CASSETTE FILE FOR WRITING.

        LDA     #OPEN           ; SET UP FOR DEVICE "OPEN."
        STA     ICCOM,X

        LDA     #OPNOT          ; DIRECTION IS "OUTPUT."
        STA     ICAX1,X
        LDA     #$80            ; SELECT SHORT IRG.
        STA     ICAX2,X

        LDA     #CFILE          ; SET UP POINTER TO DEVICE NAME.
        STA     ICBAL,X
        LDA     #CFILE/256
        STA     ICBAH,X

        JSR     CIOV            ; ATTEMPT TO OPEN FILE.
        BMI     CERR            ; ERROR.

; NOW WRITE THE ENTIRE FILE AS ONE OPERATION.
```

```
        LDA     #PUTCHR         ; SET UP FOR "PUT CHARACTERS."
        STA     ICCOM,X

        LDA     #PST            ; POINT TO START OF APPLIC. PROG.
        STA     ICBAL,X
        LDA     #PST/256
        STA     ICBAH,X

        LDA     #FLEN           ; SET UP # OF BYTES TO WRITE.
        STA     ICBLL,X
        LDA     #FLEN/256
        STA     ICBLH,X

        JSR     CIOV            ; WRITE ENTIRE FILE.
        BMI     CERR            ; ERROR.

; NOW CLOSE THE FILE AFTER SUCCESSFUL WRITE.

        LDA     #CLOSE          ; SET UP FOR "CLOSE."
        STA     ICCOM,X

        JSR     CIOV            ; CLOSE THE FILE.
        BMI     CERR            ; ERROR.

        BRK                     ; STOP WHEN DONE.

CERR    BRK                     ; STOP ON ERROR.

CFILE   .BYTE   "C:",CR         ; FILE NAME.

; THIS IS THE CARTRIDGE HEADER

*=      $BFF9                   ; "A" CARTRIDGE.

INIT    RTS
        .WORD   BOOTB
        .BYTE   0,4

        .WORD   INIT
        .END
```

11 ADVANCED TECHNIQUES AND APPLICATION NOTES

This section presents information to use the capabilities of the OS and some of the hardware capabilites that aren't directly available through the OS ,and in fact, can be in direct conflict with parts of the OS.

SOUND GENERATION

The OS uses the POKEY sound generation capabilities only in the I/O subsystem, for cassette FSK tone generation, and for the "noisy bus" option in SIO.

Capabilities

The hardware provides four independently programmable audio channels that are mixed and sent to the television set as part of the composite video signal. The POKEY registers shown below are all concerned with sound control (as described in the ATARI Home Computer Hardware Manual).

AUDCTL [D208]		Audio control.
AUDC1 [D201]	and AUDF1 [D200]	Channel 1 control.
AUDC2 [D203]	and AUDF2 [D202]	Channel 2 control.
AUDC3 [D205]	and AUDF3 [D204]	Channel 3 control.
AUDC4 [D207]	and AUDF4 [D206]	Channel 4 control.

Conflicts With OS

There are two potential conflicts with the OS involving sound generation:

- The OS can generate its own sounds and then turn off all sounds as part of I/O operations to the cassette and the serial bus peripherals.
- The OS does not turn off sounds when you press [SYSTEM.RESET] or [BREAK]. If the sounds are to be turned off under those conditions, the controlling program must provide that capability.

SCREEN GRAPHICS

Hardware Capabilities

The hardware capabilities for screen presentations are quite versatile; the OS uses a very small amount of the capability provided. The means of extension, however, are non-trivial; and making changes to a screen format while still utilizing the resident Display Handler will be difficult. See the ATARI Home Computer Hardware Manual for information regarding screen presentations.

OS Capabilities

The resident Display Handler arbitrarily supports 8 of the 11 possible full screen modes (11 of 14 modes if the GTIA chip is used in place of the CTIA). The resident Display Handler allows for an optional "split-screen" text window of fixed size. The hardware allows for many more options than the Display Handler supports, as will be seen by reading the ATARI Home Computer Hardware Manual.

Cursor Control

You can control the Display Handler text and graphics cursors directly (see Section 5 and Appendix L, B1-4).

Color Control

You can alter the color register assignments that the Display Handler makes upon all OPEN commands (see Appendix L B7-8 and elsewhere). Note that every system reset or Display Handler OPEN will reset the values back to the system default.

Alternate Character Sets

Two character sets are available in screen text modes 1 and 2. The value stored in the data base variable CHBAS [02F4] selects the character set of interest to you. The default value of $E0 provides capital (uppercase) letters, numbers and the punctuation characters corresponding to display codes $20 through $5F in Appendix E). The alternate value of $E2 provides lowercase letters and the special character graphics set (corresponding to display codes $60 through $7F and $00 through $1F in Appendix E).

User-defined character sets can also be obtained for text modes 0, 1, and 2 by providing the character matrix definitions in RAM and setting CHBAS to point to those definitions. CHBAS always contains the most significant bits of the memory address of the start of the character definitions, as shown below:

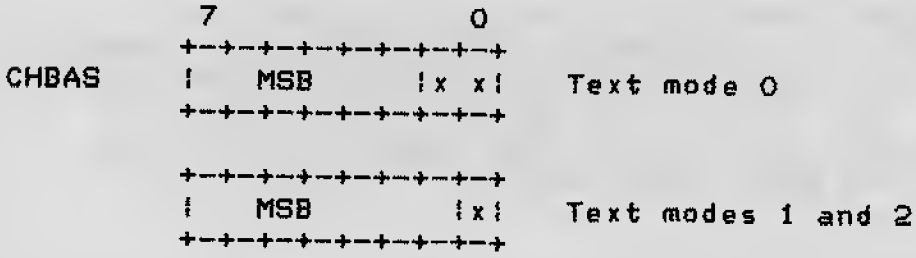


Figure 11-1 User-Defined Character Set Bit Memory Addresses

(X indicates an ignored address bit assumed to be 0.)

Each character is defined by an 8 x 8 bit matrix; the character '@' is defined as shown below:

```
        7               0
  Byte +-+-+-+-+-+-+-+-+-+
       !0!0!0!0!0!0!0!0!   0
       +-+-+-+-+-+-+-+-+-+
       !0!0!1!1!1!1!0!0!   1
       +-+-+-+-+-+-+-+-+-+
       !0!1!1!0!1!1!0!0!   2
       +-+-+-+-+-+-+-+-+-+
       !0!1!1!0!1!1!0!0!   3
       +-+-+-+-+-+-+-+-+-+
       !0!1!1!0!1!1!0!0!   4
       +-+-+-+-+-+-+-+-+-+
       !0!1!1!0!0!0!0!0!   5
       +-+-+-+-+-+-+-+-+-+
       !0!0!1!1!1!1!0!0!   6
       +-+-+-+-+-+-+-+-+-+
       !0!0!0!0!0!0!0!0!   7
       +-+-+-+-+-+-+-+-+-+
```

Figure 11-2 User Defined 8 x 8 Character Matrix Bit Table

The storage for the character set involves eight consecutive bytes for each character with characters ordered consecutively by their internal code value (see the discussion in Appendix L relating to 855).

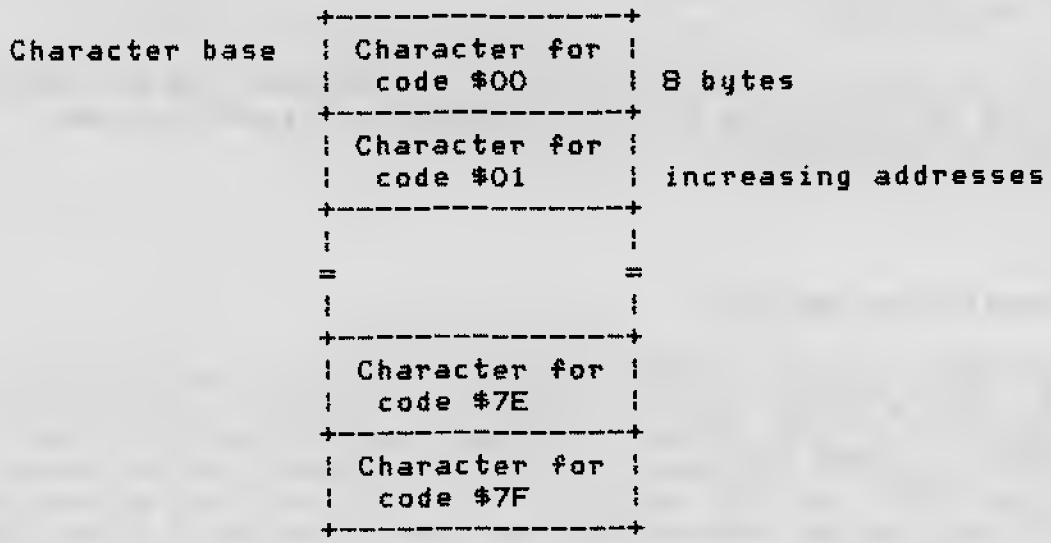


Figure 11-3 Character Base Diagram

PLAYER/MISSILE GRAPHICS

The OS makes no use of the player/missile generation capability of the hardware. It can be used independently of the OS with no conflict.

Hardware Capabilities

The hardware allows a number of independently moveable screen objects of limited width to be positioned and moved about the screen without affecting the "playfield" (bit-mapped graphics or character) data. Priority control allows the various objects to have a display precedence in case of conflict (overlap).

Conflicts With OS

You must assure that the player/missile data is address-aligned as required by PMBASE [D407]. You also must find a suitable free area that the OS guarantees to be free under all environments.

READING GAME CONTROLLERS

The OS reads the game controllers (shown below) as part of the stage 2 VBLANK process (see Appendix L J1-9):

- Joysticks/triggers 1-4.
- Paddle controllers/triggers 1-8.
- Driving controllers/triggers 1-4.
- Light pen/trigger

In addition to these controllers, other information can be sensed or sent using the PIA chip to that the console connectors are interfaced.

Keyboard Controller Sensing

Data can be read from an ATARI keyboard controller connected to the first port. This program alters registers on a chip called a PIA. To set these back to the default values to do further I/O, hit [SYSTEM.RESET] or POKE PACTL,60. If this program is to be loaded from diskette, use LOAD, not RUN and wait for the busy light on the disk drive to go out. Do not execute the program before this light goes out, otherwise the diskette continues to spin.

```
  1 GRAPHICS 0
  5 PRINT :PRINT "    KEYBOARD CONTROLLER DEMO"
 10 DIM ROW(3),I$(13),BUTTON$(1)
 30 GOSUB 6000
 40 FOR CNT=1 TO 4
 60 POSITION 2,CNT*2+5:PRINT "CONTROLLER # ";CNT;":";
```

```
70 NEXT CNT
80 FOR CNT=1 TO 4:GOSUB 7000:POSITION 19,CNT+CNT+5:PRINT BUTTON$;
   :NEXT CNT
120 GOTO 80
6000 REM ** SET UP FOR CONTROLLERS **
6010 PORTA=54016:PORTB=54017:PACTL=54018:PBCTL=54019
6020 POKE PACTL,48:POKE PORTA,255:POKE PACTL,52:POKE PORTA,221
6025 POKE PBCTL,48:POKEPORTB,255:POKE PBCTL,52:POKE PORTB,221
6030 ROW(0)=238:ROW(1)=221:ROW(2)=187:ROW(3)=119
6040 I$=" 123456789*0#"
6050 RETURN
7000 REM ** RETURN BUTTON$ WITH CHARACTER FOR BUTTON WHICH HAS
     BEEN PRESSED ON CONTROLLER CNT (1-4). **
7001 REM ** NOTE: A 1 WILL BE RETURNED IF NO CONTROLLER IS
     CONNECTED. **
7002 REM ** A SPACE WILL BE RETURNED IF THE CONTROLLER IS
     CONNECTED BUT NO KEY HAS BEEN PRESSED. **
7003 PORT=PORTA:IF CNT>2 THEN PORT=PORTB
7005 P=1
7008 PAD=CNT+CNT-2
7010 FOR J=0 TO 3
7020 POKE PORT,ROW(J)
7030 FOR I=1 TO 10:NEXT I
7050 IF PADDLE(PAD+1)>10 THEN P=J+J+J+2:GOTO 7090
7060 IF PADDLE(PAD)>10 THEN P=J+J+J+3:GOTO 7090
7070 IF STRIG(CNT-1)=0 THEN P=J+J+J+4:GOTO 7090
7080 NEXT J
7090 BUTTON$=I$(P,P)
7095 RETURN
```

Figure 11-4 Reading Data From an ATARI Keyboard Controller

The table below shows the variable/register values used for reading a keyboard controller from each of the four controller ports.

	Port 1	Port 2	Port 3	Port 4
PORT A direction bits	OF	F0	-	-
PORT B direction bits	-	-	OF	F0
Port A row select	FE,FD, FB,F7	EF,DF BF,7F	-	-
Port B row select	-	-	FE,FD, FB,F7	EF,DF, BF,7F
Column 1 Sense	PADDL1	PADDL3	PADDL5	PADDL7
Column 2 Sense	PADDL0	PADDL2	PADDL4	PADDL6
Column 3 Sense	STRIG0	STRIG1	STRIG2	STRIG3

Figure 11-5 ATARI Keyboard Controller Variable/Register Value Table

Front Panel Connectors as I/O Ports

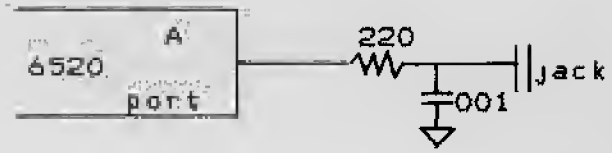
The three pages that follow show how some of the pins in the front panel (game controller) connectors can be used as general I/O pins.

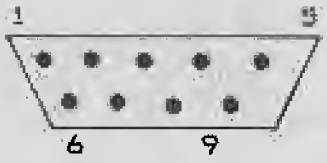
Hardware Information

PIA (6520 / 6820)
Out: TTL levels, 1 load
In : TTL levels, 1 load

For more information refer to 6520 chip manual.

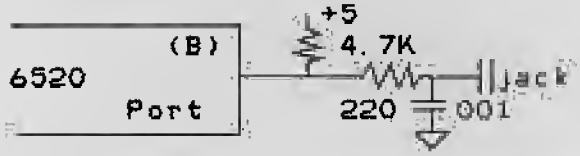
Port A Circuit (typical):





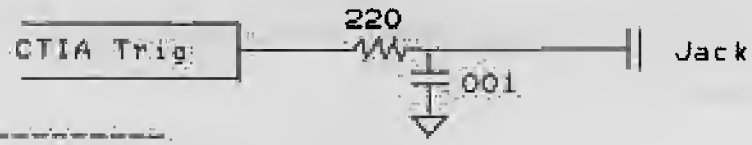
Male connector, FRONT view
Pin 8 = Ground
Pin 7 = Vcc 8+5v *)

Port B Circuit (typical):



Note: 50mA maximum
total external drain
on power supply allowed

"Trigger" Port Circuit (typical):



Software Information

6520 PIA: (This also pertains to all of the following: **)
Port A control (address D302)

Write this into this register

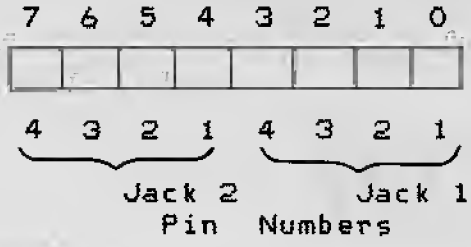
Port A Data/Data direction addres
ing control
0 = Data Direction is at D300
1 = data is at D300

Port A data direction (address D300)

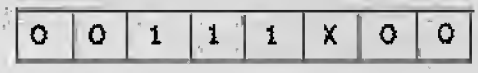
Write this into this register.

Data direction control
for Port A
1 = Out
0 = In

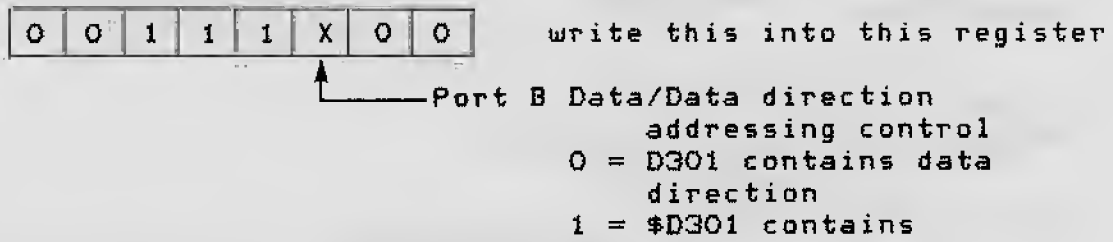
Port A data (address D300)



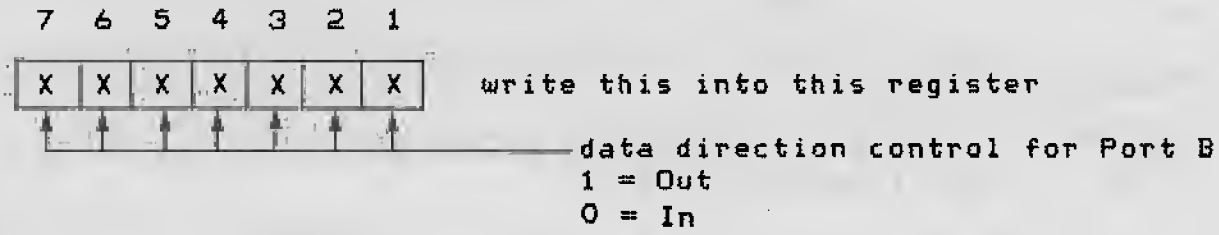
Port B Control (address D303)



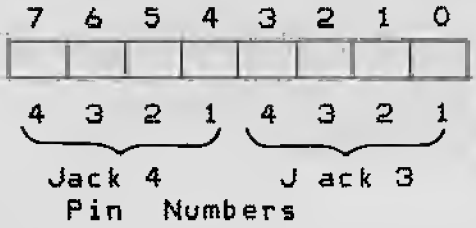
6520 PIA:
Port B Control (address D303)



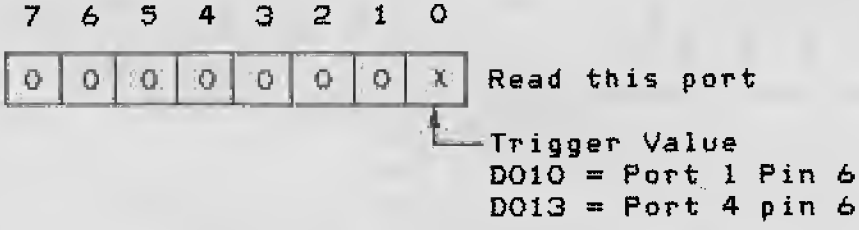
Port B data direction (address D301)



Port B data (address D301)



Four "Trigger" ports: D010, D011, D012, D013



Other Miscellaneous Software Information

1). The OS sets up all PIA ports as inputs during initialization.
2). The OS usually reads the above once per television frame (during vertical-blank) into RAM as follows:

Data Base Name	Address	Data	Pins S
STICK0	0278	7 6 5 4 3 2 1 0 0 0 0 0 X X X X	Jack 1, pins 4,3,2, if 10053.7
STICK1	0729		Jack 2, Pins 4,3,2,1
STICK2	027A		Jack 3, Pins 4,3,2,1
STICK3	027B		Jack 4, Pins 4,3,2,1
STRIG0	0284		Jack 1, Pin 6
STRIG1	0285	7 6 5 4 3 2 1 0 0 0 0 0 0 0 0	Jack 2, Pin 6
STRIG2	0286		Jack 3, Pin 6
STRIG3	0287		Jack 4, Pin 6
PADDL1	0270	7 6 5 4 3 2 1 0 X X X X X X X X	Jack 1, Pin 5
PADDL3	0272		Jack 2, Pin 5
PADDL5	0274		Jack 3, Pin 5
PADDL7	0276		Jack 4, Pin 5
PADDL0	0271		Jack 1, Pin 9
PADDL2	0273		Jack 2, Pin 9
PADDL4	0275		Jack 3, Pin 9
PADDL 6	0277		Jack 4, Pin 9

Figure 11-6 Using Front Panel Connectors As I/O Ports: Pin Function Tables

* Pins 5 and 9 are read through the paddle controller circuitry a nominal value of 7 indicates that the pin is high (or floating) and a nominal value of 228 indicates that the pin is pulled low.

Appendix A -- CIO COMMAND BYTE VALUES

The following hex values are known to be legitimate CIO commands.

Most handlers:

```
03 -- OPEN
05 -- GET RECORD
07 -- GET CHARACTERS
09 -- PUT RECORD
0B -- PUT CHARACTERS
0C -- CLOSE
0D -- GET STATUS
```

Display Handler only:

```
11 -- FILL
12 -- DRAW
```

Diskette File Manager only:

```
20 -- RENAME
21 -- DELETE
22 -- FORMAT
23 -- LOCK
24 -- UNLOCK
25 -- POINT
26 -- NOTE
```

Appendix B -- CIO STATUS BYTE VALUES

Shown below are the known CIO STATUS BYTE values.

01 (001) -- OPERATION COMPLETE (NO ERRORS)

80 (128) -- [BREAK] KEY ABORT
81 (129) -- IOCB ALREADY IN USE (OPEN)
82 (130) -- NON-EXISTENT DEVICE
83 (131) -- OPENED FOR WRITE ONLY
84 (132) -- INVALID COMMAND
85 (133) -- DEVICE OR FILE NOT OPEN
86 (134) -- INVALID IOCB NUMBER (Y reg only)
87 (135) -- OPENED FOR READ ONLY
88 (136) -- END OF FILE
89 (137) -- TRUNCATED RECORD
8A (138) -- DEVICE TIMEOUT (DOESN'T RESPOND)
8B (139) -- DEVICE NAK
8C (140) -- SERIAL BUS INPUT FRAMING ERROR
8D (141) -- CURSOR out-of-range
8E (142) -- SERIAL BUS DATA FRAME OVERRUN ERROR
8F (143) -- SERIAL BUS DATA FRAME CHECKSUM ERROR
90 (144) -- DEVICE DONE ERROR
91 (145) -- BAD SCREEN MODE
92 (146) -- FUNCTION NOT SUPPORTED BY HANDLER
93 (147) -- INSUFFICIENT MEMORY FOR SCREEN MODE

A0 (160) -- DISK DRIVE # ERROR
A1 (161) -- TOO MANY OPEN DISK FILES
A2 (162) -- DISK FULL
A3 (163) -- FATAL DISK I/O ERROR
A4 (164) -- INTERNAL FILE # MISMATCH
A5 (165) -- FILE NAME ERROR
A6 (166) -- POINT DATA LENGTH ERROR
A7 (167) -- FILE LOCKED
A8 (168) -- COMMAND INVALID FOR DISK
A9 (169) -- DIRECTORY FULL (64 FILES)
AA (170) -- FILE NOT FOUND
AB (171) -- POINT INVALID

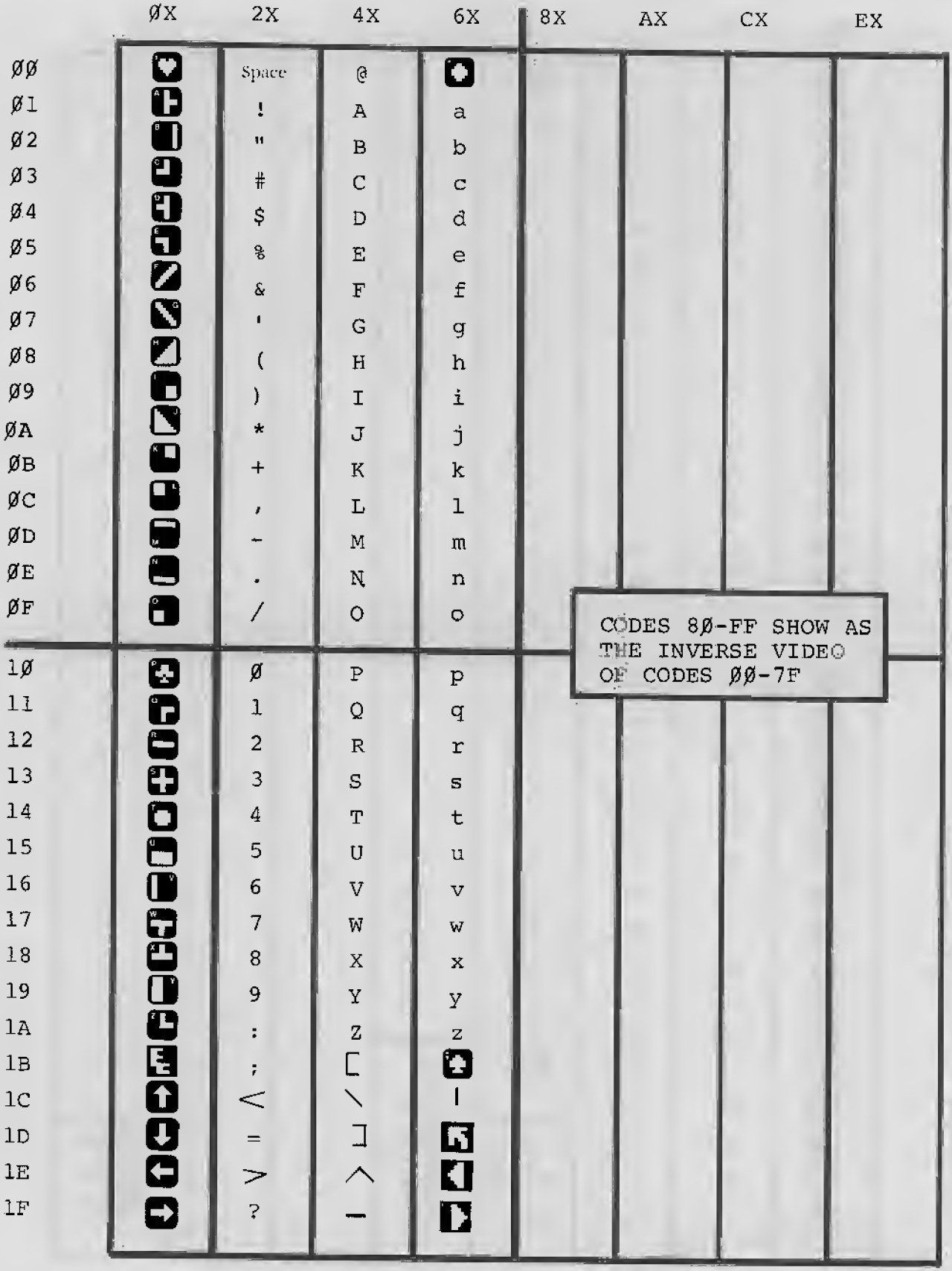
Appendix C -- SIO STATUS BYTE VALUES

Shown below are the known SIO STATUS BYTE hexadecimal values.

01 (001) -- OPERATION COMPLETE (NO ERRORS)

8A (138) -- DEVICE TIMEOUT (DOESN'T RESPOND)
8B (139) -- DEVICE NAK
8C (140) -- SERIAL BUS INPUT FRAMING ERROR
8E (142) -- SERIAL BUS DATA FRAME OVERRUN ERROR
8F (143) -- SERIAL BUS DATA FRAME CHECKSUM ERROR
90 (144) -- DEVICE DONE ERROR

	ØX	2X	4X	6X	8X	AX	CX	EX
ØØ		Space	@					
Ø1		!	A	a				
Ø2		"	B	b				
Ø3		#	C	c				
Ø4		$	D	d				
Ø5		%	E	e				
Ø6		&	F	f				
Ø7		'	G	g				
Ø8		(	H	h				
Ø9		)	I	i				
ØA		*	J	j				
ØB		+	K	k				
ØC		,	L	l				
ØD		-	M	m				
ØE		.	N	n				
ØF		/	O	o				
1Ø		Ø	P	p				
11		1	Q	q				
12		2	R	r				
13		3	S	s				
14		4	T	t				
15		5	U	u				
16		6	V	v				
17		7	W	w				
18		8	X	x				
19		9	Y	y				
1A		:	Z	z				
1B	ESC	;			EOL			
1C		<	\	|	DEL LINE			
1D		=		CLEAR	INS LINE			BELL
1E		>	^	BACKSP	CLR TAB			DEL CHAR
1F		?	_	TAB	SET TAB			INS CHAR


ØX
2X
4X
6X
8X
AX
CX
EX
ØØ
Space
@
Ø1
!
A
a
Ø2
"
B
b
Ø3
#
C
c
Ø4
$
D
d
Ø5
%
E
e
Ø6
&
F
f
Ø7
'
G
g
Ø8
(
H
h
Ø9
)
I
i
ØA
*
J
j
ØB
+
K
k
ØC
,
L
l
ØD
-
M
m
ØE
.
N
n
ØF
/
O
o
CODES 8Ø-FF SHOW AS
THE INVERSE VIDEO
OF CODES ØØ-7F
1Ø
Ø
P
p
11
1
Q
q
12
2
R
r
13
3
S
s
14
4
T
t
15
5
U
u
16
6
V
v
17
7
W
w
18
8
X
x
19
9
Y
y
1A
:
Z
z
1B
;
1C
=
1D
1E
1F
?

Appendix F -- KEYBOARD CODES (ATASCII)

CTRL			SHIFT & LOWER			SHIFT			LOWER		
00	,	20	20	<space>	21	40	@	35	60	^.	22
01	A	3F	21	!	1F	41	A	3F	61	a	3F
02	B	15	22	"	1E	42	B	15	62	b	15
03	C	12	23	#	1A	43	C	12	63	c	12
04	D	3A	24	$	18	44	D	3A	64	d	3A
05	E	2A	25	%	1D	45	E	2A	65	e	2A
06	F	38	26	&	1B	46	F	38	66	f	38
07	G	3D	27	'	33	47	G	3D	67	g	3D
08	H	39	28	(	30	48	H	39	68	h	39
09	I	0D	29	)	32	49	I	0D	69	i	0D
0A	J	01	2A	*	07	4A	J	01	6A	j	01
0B	K	05	2B	+	06	4B	K	05	6B	k	05
0C	L	00	2C	,	20	4C	L	00	6C	l	00
0D	M	25	2D	-	0E	4D	M	25	6D	m	25
0E	N	23	2E	.	22	4E	N	23	6E	n	23
0F	O	08	2F	/	26	4F	O	08	6F	o	08
10	P	0A	30	0	32	50	P	0A	70	p	0A
11	Q	2F	31	1	1F	51	Q	2F	71	q	2F
12	R	28	32	2	1E	52	R	28	72	r	28
13	S	3E	33	3	1A	53	S	3E	73	s	3E
14	T	2D	34	4	18	54	T	2D	74	t	2D
15	U	0B	35	5	1D	55	U	0B	75	u	0B
16	V	10	36	6	1B	56	V	10	76	v	10
17	W	2E	37	7	33	57	W	2E	77	w	2E
18	X	16	38	8	35	58	X	16	78	x	16
19	Y	2B	39	9	30	59	Y	2B	79	y	2B
1A	Z	17	3A	:	02	5A	Z	17	7A	z	17
1B	<esc>	1C	3B	;	0D	5B	[	20	7B	;	02
1C	^<up>	0E	3C	<	36	5C	\	06	7C	!	0F
1D	^<down>	0F	3D	=	0F	5D	]	22	7D	<clear>	36
1E	^<left>	06	3E	>	37	5E	^	07	7E	<back>	34
1F	^<right>	07	3F	?	26	5F	_	0E	7F	<tab>	2C

80-9A	/!\	00-1A	9F	s<tab>	2C
9B	<return>	and ^3 0C,1A	A0-FC	/!\	20-7C
9C	s<del>	34	FD	^2	1E
9D	s<insert>	37	FE	^<del>	34
9E	^<tab>	2C	FF	^<insert>	37

```
<clear> ::= s< or  ^<
<return> ::= <return> or s<return> or  ^<return>
<esc> ::= <esc> or s<esc> or  ^<esc>
<space> ::= <space> or s<space> or  ^<space>

Where: s as a prefix indicates [SHIFT].
       ^ as a prefix indicates [CTRL].
       /!\ as a prefix indicates ATARI key inverse active.
```

Appendix G -- PRINTER CODES (ATASCII)

Character set for "normal" mode printing:

20	<space>	40	@	60	`
21	!	41	A	61	a
22	"	42	B	62	b
23	#	43	C	63	c
24	$	44	D	64	d
25	%	45	E	65	e
26	&	46	F	66	f
27	'	47	G	67	g
28	(	48	H	68	h
29	)	49	I	69	i
2A	*	4A	J	6A	j
2B	+	4B	K	6B	k
2C	,	4C	L	6C	l
2D	-	4D	M	6D	m
2E	.	4E	N	6E	n
2F	/	4F	O	6F	o
30	0	50	P	70	p
31	1	51	Q	71	q
32	2	52	R	72	r
33	3	53	S	73	s
34	4	54	T	74	t
35	5	55	U	75	u
36	6	56	V	76	v
37	7	57	W	77	w
38	8	58	X	78	x
39	9	59	Y	79	y
3A	:	5A	Z	7A	z
3B	;	5B	[	7B	{
3C	<	5C	\	7C	¦
3D	=	5D	]	7D	}
3E	>	5E	^	7E	~
3F	?	5F	_	7F	<space>

Note: The following codes print differently than defined by the ATASCII definition.

- 00 through 1F print blank.
- 60 prints ` instead of "diamond".
- 7B prints { instead of "spade".
- 7D prints } instead of "clear".
- 7E prints ~ instead of "backspace".
- 7F prints blank instead of "tab".

Character set for "sideways" mode printing:

		40	@	60	@
		41	A	61	A
		42	B	62	B
		43	C	63	C
		44	D	64	D
		45	E	65	E
		46	F	66	F
		47	G	67	G
		48	H	68	H
		49	I	69	I
		4A	J	6A	J
		4B	K	6B	K
		4C	L	6C	L
		4D	M	6D	M
		4E	N	6E	N
		4F	O	6F	O
30	0	50	P	70	P
31	1	51	Q	71	Q
32	2	52	R	72	R
33	3	53	S	73	S
34	4	54	T	74	T
35	5	55	U	75	U
36	6	56	V	76	V
37	7	57	W	77	W
38	8	58	X	78	X
39	9	59	Y	79	Y
3A	:	5A	Z	7A	Z
3B	;	5B	[	7B	[
3C	<	5C	\	7C	\
3D	=	5D	]	7D	]
3E	>	5E	<up>	7E	<up>
3F	?	5F	<left>	7F	<left>

Note: the following codes print differently than defined by the ATASCII definition.

00 through 2F print blank.
5E prints "up arrow" instead of ^.
5F prints "left arrow" instead of _.
60 through 7F repeats 40 through 5F instead of proper set.

Appendix H -- SCREEN MODE CHARACTERISTICS

Mode #	Horiz. Posit.	Vert. W/O Sp	Vert. W Sp	Colors	Data Value	Color Reg.	Memory Reqd. (split)	(full)
0	40	24	--	2	backgd.	BAK	992	992
					00-FF	PF 2		
					"	PF 1*		
1	20	24	20	5	backgd.	BAK	674	672
					00-3F	PF 0		
					40-7F	PF 1		
					80-BF	PF 2		
					C0-FF	PF 3		
2	20	12	10	5	backgd.	BAK	424	420
					00-3F	PF 0		
					40-7F	PF 1		
					80-BF	PF 2		
					C0-FF	PF 3		
3	40	24	20	4	0	BAK	434	432
					1	PF 0		
					2	PF 1		
					3	PF 2		
4	80	48	40	2	0	BAK	694	696
					1	PF 0		
5	80	48	40	4	0	BAK	1174	1176
					1	PF 0		
					2	PF 1		
					3	PF 2		
6	160	96	80	2	0	BAK	2174	2184
					1	PF 0		
7	160	96	80	4	0	BAK	4190	4200
					1	PF 0		
					2	PF 1		
					3	PF 2		
8	320	192	160	2	0	PF 2	8112	8138
					1	PF 1*		
9	80	192	--	1	Note 2			8138
10	80	192	--	9	0	PM 0		8138
					1	PM 1		
					2	PM 2		
					3	PM 3		
					4	PF 0		

				5	PF 1	
				6	PF 2	
				7	PF 3	
				8	BAK	
				9	BAK	
				A	BAK	
				B	BAK	
				C	PF 0	
				D	PF 1	
				E	PF 2	
				F	PF 3	
11	80	192	--	16	Note 3	8138

Notes:

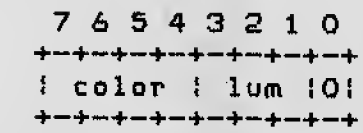
* Uses color of PF 2, lum of PF 1.
2 Uses color of BAK, lum of data value ($0-F).
3 Uses color of data value ($0-F), lum of BAK.

PF x ::= Playfield color register x.
PM x ::= Player/Missile Graphics color register x.
BAK ::= Background color register (also known as PF 4).

The default values for the color registers are shown below:

BAK	=	$00
PF0	=	$28
PF1	=	$CA
PF2	=	$94
PF3	=	$46

The form of a color register byte is shown below:



Where:

color (hex values)	lum
0 = gray	0 = minimum luminance
1 = light orange	1 = ¦
2 = orange	2 = ¦
3 = red orange	3 = (increasing
4 = pink	4 = luminance)
5 = purple	5 = ¦
6 = purple-blue	6 = ¦
7 = blue	7 = maximum luminance
8 = blue	
9 = light blue	
A = turquoise	
B = green-blue	
C = green	
D = yellow-green	
E = orange-green	
F = light orange	

Appendix I -- SERIAL BUS ID AND COMMAND SUMMARY

Serial bus device IDs

Floppy diskettes	D1-D4	$31-34
Printer	P1	$40
RS-232-C	R1-R4	$50-53

Serial bus control codes

ACK	- $41	('A')
NAK	- $4E	('N')
COMPLETE	- $43	('C')
ERR	- $45	('E')

Serial bus command codes

READ	- $52	('R')	Disk
WRITE	- $57	('W')	Printer/Disk
STATUS	- $53	('S')	Printer/Disk
PUT(no check)	- $50	('P')	Disk
FORMAT	- $21	('!')	Disk
READ ADDRESS	- $54	('T')	
READ SPIN	- $51	('Q')	Disk
MOTOR ON	- $55	('U')	Disk
VERIFY SECTOR	- $56	('V')	Disk

Appendix J -- ROM VECTORS

The fixed address OS ROM JMP vectors are shown below; at each address is a JMP instruction to the indicated routine.

Name	Addr	Reference	Function
DISKIV	E450	*	Diskette Handler initialization
DSKINV	E453	5.4.2	Diskette Handler entry.
CIOV	E456	5.2	CIO utility entry.
SIOV	E459	9.3	SIO utility entry.
SETVBV	E45C	6.7.2	Set System Timers routine.
SYSVBV	E45F	6.3	Stage 1 VBLANK entry.
XITVBV	E462	6.3	Exit VBLANK entry.
SIOINV	E465	*	SIO utility initialization.
SENDEV	E468	*	Send enable routine.
INTINV	E46B	*	Interrupt Handler initialization.
CIOINV	E46E	*	CIO utility initialization.
BLKBDV	E471	3.1.1	Blackboard mode entry.
WARMSV	E474	7.	Warmstart ([SYSTEM.RESET]) entry.
COLDSV	E477	7.	Coldstart (power-up) entry.
RBLOKV	E47A	*	Cassette-read block entry.
CSOPIV	E47D	*	Cassette-OPEN input entry.

* These vectors are for OS internal use only.

The fixed address Floating Point Package ROM routine entry point addresses are shown below; complete descriptions of the corresponding routines are provided in Section 8.

AFP	D800	ASCII to FP convert.
FASC	D8E6	FP to ASCII convert.
IFP	D9AA	Integer to FP convert.
FPI	D9D2	FP to integer convert.
FADD	DA66	FP add.
FSUB	DA60	FP subtract.
FMUL	DADB	FP multiply
FDIV	DB28	FP divide.
LOG	DECD	FP base e logarithm.
LOG10	DED1	FP base 10 logarithm.
EXP	DDC0	FP base e exponentiation.
EXP10	DDCC	FP base 10 exponentiation.
PLYEVL	DD40	FP polynomial evaluation.
ZFR0	DA44	Clear FR0.
ZF1	DA46	Clear FP number.
FLD0R	DD89	Load FP number.
FLD0P	DD8D	Load FP number.
FLD1R	DD98	Load FP number.
FLD1P	DD9C	Load FP number.
FSTOR	DDA7	Store FP number.
FSTOP	DDAB	Store FP number.
FMOVE	DDB6	Move FP number.

The base addresses of the Handler vectors for the resident handlers are shown below:

Screen Editor (E)	E400
Display Handler (S)	E410
Keyboard Handler (K)	E420
Printer Handler (P)	E430
Cassette Handler (C)	E440

See Section 5 for the format of the entry for each Handler.

The 6502 Computer interrupt vector values are shown below:

Function	Address	Value
NMI	FFFA	E7B4
RESET	FFAC	E477
IRQ	FFFE	E6FE

Appendix K -- DEVICE CHARACTERISTICS

This appendix describes the physical characteristics of the devices that interface to the ATARI 400 and ATARI 800 Home Computers. Where applicable, data capacity, data transfer rate, storage format, SIO interface, and cabling will be detailed.

KEYBOARD

The keyboard input rate is limited by the OS keyboard reading procedure to be 60 characters per second. The code for each key is shown in Table 5-4. The keyboard hardware has no buffering and is rate-limited by the debounce algorithm used.

DISPLAY

The television screen display generator has many capabilities that are not used by the Display Handler (as described in Section 5 and shown in Appendix H). There are additional display modes, object generators, hardware display scrolling, and many other features that are described in the ATARI Home Computer Hardware Manual.

Since all display data is stored in RAM, the display data update rate is limited primarily by the software routines that generate and format the data and access the RAM. The generation of the display from the RAM is accomplished by the ANTIC and CTIA or GTIA chips using Direct Memory Access (DMA) to access the RAM data.

The internal storage formats for display data for the various modes are detailed in the ATARI Home Computer Hardware Manual.

ATARI 410 PROGRAM RECORDER

The ATARI 410 Program Recorder has the following characteristics:

DATA CAPACITY:

    100 characters per C-60 tape (unformatted).

DATA TRANSFER RATES:

    * 600 Baud (60 characters per second)

*Note: The OS has the ability to adjust to different tape speeds (447 - 895 Baud).

STORAGE FORMAT:

Tapes are recorded in 1/4 track stereo format at 1 7/8 inches per second. The tape can be recorded in both directions, where tracks 1 and 2 are side A left and right; and tracks 3 and 4 are side B right and left (industry standard). On each side, the left channel (1 or 4) is used for audio and the right channel (2 and 3) is used for digital information.

The audio channel is recorded the normal way. The digital channel is recorded using the POKEY two-tone mode producing FSK data at up to 600 baud. The MARK frequency is 5327 Hz and the SPACE frequency is 3995 Hz. The transmission of data is asynchronous byte serial as seen from the computer; POKEY reads or writes a bit serial FSK sequence for each byte, in the following order:

```
1 start bit (SPACE)
data bit 0 -+
data bit 1  |
.           +-  0 = SPACE, 1 = MARK.
data bit 6  |
data bit 7 -+
1 stop bit (MARK)
```

The only control the computer has over tape motion is motor start/stop; and this only if the PLAY button is pressed by the user. In order for recording to take place, the user must press both the REC and PLAY buttons on the cassette. The computer has no way to sense the position of these buttons, nor even if an ATARI 410 Program Recorder is cabled to the computer, so the user must be careful when using this device.

SIO INTERFACE

The cassette device utilizes portions of the serial bus hardware, but does not follow any of the protocol as defined in Section 9.

ATARI 820[TM] 40-COLUMN IMPACT PRINTER

The ATARI 820 Printer has the following characteristics:

DATA CAPACITY:

40 characters per line (normal printing)
29 characters per line (sideways printing)

DATA TRANSFER RATES:

Bus rate: xx characters per second.
Print time (burst): xx characters per second.
Print time (average): xx characters per second.

STORAGE FORMAT:

3 7/8 inch wide paper.
5X7 dot matrix, impact printing.

Normal format --
40 characters per line.
6 lines per inch (vertical).
12 characters per inch (horizontal).

Sideways format --
29 characters per line.
6 lines per inch (vertical).
9 characters per inch (horizontal).

SIO INTERFACE

The controller serial bus ID is $40.

The controller supports the following SIO commands (see Section 5 for more information regarding the Handler and Section 9 for a general discussion of bus commands):

GET STATUS

The computer sends a command frame of the format shown below:

Device ID = $40.
Command byte = $53.
auxiliary 1 = doesn't matter.
auxiliary 2 = doesn't matter.
Checksum = checksum of bytes above.

The printer controller responds with a data frame of the format shown earlier in this appendix as part of the GET STATUS discussion.

PRINT LINE

The computer sends a command frame of the format shown below:

Device ID = $40.
Command byte = $57.

auxiliary 1 = doesn't matter.
auxiliary 2 = $4E for normal print or $53 for sideways.
Checksum = checksum of bytes above.

The computer sends a data frame of the format shown below:

Leftmost character of line (column 1).
Next character of line (column 2).
.
.
Rightmost character of line (column 40 or 29).
Checksum byte.

Note that the data frame size is variable, either 41 or 30 bytes in length, depending upon the print mode specified in the command frame.

ATARI 810 DISK DRIVE

The ATARI 810[TM] Disk Drive has the following characteristics:

DATA CAPACITY:

720 sectors of 128 bytes each (Disk Handler format).
709 sectors of 125 data bytes each (Disk File Manager format).

DATA TRANSFER RATES:

Bus rate: 1920 characters per second.
Seek time: 5.25 msec. per track + 10 to 210 msec.
Rotational latency: 104 msec maximum (288 rpm).

STORAGE FORMAT:

5 1/4 inch diskette, soft sectored by the controller.
40 tracks per diskette.
18 sectors per track.
128 bytes per sector.
Controlled by National INS1771-1 formatter/controller chip.
Sector sequence per track is: 18, 1, 3, 5, 7, 9, 11, 13, 15,
17, 2, 4, 6, 8, 10, 12, 14, 16

SIO INTERFACE

The controller serial bus IDs range from $31 (for 'D1') to $34 (for 'D4').

The controller supports the following SIO commands (see earlier in this Appendix for information about the Diskette Handler and Section 9 for a general discussion of bus commands):

GET STATUS

The computer sends a command frame of the format shown below:

```
Device ID = $31-34.
Command byte = $53.
auxiliary 1 = doesn't matter.
auxiliary 2 = doesn't matter.
Checksum     = checksum of bytes above.
```

The diskette controller responds with a data frame of the format shown earlier in this Appendix as part of the STATUS REQUEST discussion.

PUT SECTOR (WITH VERIFY)

The computer sends a command frame of the format shown below:

```
Device ID = $31-34
Command byte = $57.
auxiliary 1 = low byte of sector number.
auxiliary 2 = high byte of sector number (1-720).
Checksum     = checksum of bytes above.
```

The computer sends a data frame of the format shown below:

```
128 data bytes.
Checksum byte.
```

The diskette controller writes the frame data to the specified sector, then reads the sector and compares the content with the frame data. The COMPLETE byte value indicates the status of the operation.

PUT SECTOR (NO VERIFY)

The computer sends a command frame of the format shown below:

```
Device ID = $31-34
Command byte = $50.
auxiliary 1 = low byte of sector number.
auxiliary 2 = high byte of sector number (1-720).
Checksum     = checksum of bytes above.
```

The computer sends a data frame of the format shown below:

128 data bytes.
Checksum byte.

The diskette controller writes the frame data to the specified sector, then sends a COMPLETE byte value that indicates the status of the operation.

GET SECTOR

The computer sends a command frame of the format shown below:

Device ID = $31-34
Command byte = $52.
auxiliary 1 = low byte of sector number.
auxiliary 2 = high byte of sector number (1-720).
Checksum = checksum of bytes above.

The diskette controller sends a data frame of the format shown below:

128 data bytes.
Checksum byte.

FORMAT DISKETTE

The computer sends a command frame of the format shown below:

Device ID = $31-34
Command byte = $21.
auxiliary 1 = doesn't matter.
auxiliary 2 = doesn't matter.
Checksum = checksum of bytes above.

The diskette controller completely formats the diskette (generates 40 tracks of 18 soft sectors per track with the data portion of each sector equal to all zeros) and then reads each sector to verify its integrity. A data frame of 128 bytes plus checksum is returned in that the sector numbers of all bad sectors (up to a maximum of 63 sectors) are contained, followed by two consecutive bytes of $FF. If there are no bad sectors on the diskette the first 2 bytes of the data

Appendix L -- OS DATA BASE VARIABLE FUNCTIONAL DESCRIPTIONS

CENTRAL DATA BASE DESCRIPTION

This appendix provides detailed information for those variables in the OS data base that can be altered by the user. Remaining variables are provided narrative descriptions. Information on the variables is presented in a multiple access scheme: Lookup tables are referenced to a common set of narratives, that is itself ordered by function.

Variable descriptions are referenced by a label called a variable identifier (VID) number. The label comprises a single letter followed by a number. A different letter is assigned for each major functional area being described, and the numbers are assigned sequentially within each functional area. Those variables that are not considered to be of interest to any user are flagged with an asterisk (*) after their names. The data base lookup tables provided are:

1. Functional grouping -- index to the function narrative and descriptions of variables, giving VID and variable name.

2. Alphabetic list of names -- giving VID of description.

3. Address ordered list -- giving VID of description.

Item 1, the functional grouping index, starts on the next page; the other two lookup tables are at the end of Appendix L.

FUNCTIONAL INDEX TO DATA BASE VARIABLE DESCRIPTIONS

A. Memory configuration
   - A1 MEMLO
   - A2 MEMTOP
   - A3 APPMHI
   - A4 RAMTOP
   - A5 RAMSIZ

B. Text/graphics screen

   Cursor control
   - B1 CRSINH
   - B2 ROWCRS, COLCRS
   - B3 OLDROW, OLDCOL
   - B4 TXTROW, TXTCOL

   Screen margins
   - B5 LMARGN
   - B6 RMARGN

   Color control
   - B7 PCOLR0 - PCOLR3
   - B8 COLOR0 - COLOR4

   Text scrolling
   - B9 SCRFLG*

   Attract mode
   - B10 ATRACT
   - B11 COLRSH*
   - B12 DRKMSK*

   Tabbing
   - B13 TABMAP

   Logical text lines
   - B14 LOGMAP*
   - B15 LOGCOL*

   Split screen

```
  B16 BOTSCR*

FILL/DRAW function
  B17 FILDAT
  B18 FILFLG*
  B19 NEW 5ROW*, NEWCOL*
  B20 HOLD4*



  B21 ROWINC*, COLINC*
  B22 DELTAR*, DELTAC*
  B23 COUNTR*
  B24 ROWAC*, COLAC*
  B25 ENDPT*

Displaying control characters

  Escape (display following control char)
    B26 ESCFLG*

  Display control characters mode
    B27 DSPFLG

Bit mapped graphics
  B28 DMASK*
  B29 SHFAMT*
```

```
Internal working variables
  B30 HOLD1*
  B31 HOLD2*
  B32 HOLD3*
  B33 TMPCHR*
  B34 DSTAT*
  B35 DINDEX
  B36 SAVMSC
  B37 OLDCHR*
  B38 OLDADR*
  B39 ADRESS*
  B40 MLTTMP/OPNTMP/TOADR*
  B41 SAVADR/FRMADR*
  B42 BUFCNT*
  B43 BUFSTR*
  B44 SWPFLG*
  B45 INSDAT*
  B46 TMPROW*,TMPCOL*
  B47 TMPLBT*
  B48 SUBTMP*
  B49 TINDEX*
  B50 BITMSK*
  B51 LINBUF*
  B52 TXTMSC
  B53 TXTOLD*
```

Internal character code conversion
  B54 ATACHR
  B55 CHAR*

C. Disk Handler
  C1 BUFADR*
  C2 DSKTIM*

D. Cassette (part in SIO part in Handler)

Baud rate determination
  D1 CBAUDL*, CBAUDH*
  D2 TIMFLG*
  D3 TIMER1*, TIMER2*
  D4 ADDCOR*
  D5 TEMP1*
  D6 TEMP3*
  D7 SAVIO*

Cassette mode
  D8 CASFLG*

Cassette buffer
  D9 CASBUF*
  D10 BLIM*
  D11 BPTR*

Internal working variables
  D12 FEOF*
  D13 FTYPE*
  D14 WMODE*
  D15 FREQ*

E. Keyboard

Key reading and debouncing
  E1 CH1*
  E2 KEYDEL*
  E3 CH

Special functions

Start/stop
E4 SSFLAG

[BREAK]
E5 BRKKEY

[SHIFT]/[CONTROL] lock
E6 SHFLOK
E7 HOLDCH*

Autorepeat
E8 SRTIMR*

Inverse video
E9 INVFLG

Console switches ([SELECT], [START], and [OPTION])

F. Printer

printer-buffer

F1 PRNBUF*
F2 PBUFSZ*
F3 PBPNT*

Internal working variables
F4 PTEMP*
F5 PTIMOT*

G. Central I/O routine (CIO)

User call parameters

G1 IOCB
G2 ICHID
G3 ICDNO
G4 ICCOM
G5 ICSTA
G6 ICBAL, ICBAH
G7 ICPTL, ICPTH
G8 ICBLL, ICBLH
G9 ICAX1, ICAX2
G10 ICSPR

Device status
G11 DVSTAT

device table
G12 HATABS

```
   CIO/Handler interface Parameters

     G13 ZIOCB (IOCBAS)
       G14 ICHIDZ
       G15 ICDNOZ
       G16 ICCOMZ
       G17 ICSTAZ
       G18 ICBALZ,ICBALH
       G19 ICPTLZ,ICPTHZ
       G20 ICBLLZ,ICBLHZ
       G21 ICAX1Z,ICAX2Z
       G22 ICSPRZ (ICIDNO,CIOCHR)

   Internal working variables
     G23 ICCOMT*
     G24 ICIDNO*
     G25 CIOCHR*


H. Serial I/O routine (SIO)


   User call parameters

     H1 DCB control block
       H2 DDEVIC
       H3 DUNIT
       H4 DCOMND
       H5 DSTATS
       H6 DBUFLO,DBUFHI
       H7 DTIMLO
       H8 DBYTLO,DBYTHI
       H9 DAUX1,DAUX2

     Bus sound control
       H10 SOUNDR

   Serial bus control

     Retry logic
       H11 CRETRY*
       H12 DRETRY*

     Checksum
       H13 CHKSUM*
       H14 CHKSNT*
       H15 NOCKSM*
```

Data buffering

General buffer control

H16 BUFRLO*,BUFRHI*
H17 BFENLO*,BFENHI*

Command frame output buffer
H18 CDEVIC*
H19 CCOMND*
H20 CAUX1*,CAUX2*

Receive/transmit data buffering
H21 BUFRFL*
H22 RECVDN*
H23 TEMP*
H24 XMTDON*

SIO timeout
H25 TIMFLG*
H26 CDTMV1*
H27 CDTMA1*

Internal working variables
H28 STACKP*
H29 TSTAT*
H30 ERRFLG*
H31 STATUS*
H32 SSKCTL*

J. ATARI controllers

Joysticks
J1 STICK0 - STICK3
J2 STRIG0 - STRIG3

Paddles
J3 PADDL0 - PADDL7
J4 PTRIG0 - PTRIG7

Paddle controllers
J8 STICK0 - STICK3
J9 STRIG0 - STRIG3

K. Disk file manager
K1 FMSZPG*
K2 ZBUFP*
K3 ZDRVA*
K4 ZSBA*
K5 ERRNO*

L. Disk utilities (DOS)
   L1 DSKUTL*

M. Floating point package
   M1 FRO
   M2 FRE*
   M3 FR1
   M4 FR2*
   M5 FRX*
   M6 EEXP*
   M7 NSIGN*
   M8 ESIGN*
   M9 FCHRFLG*
   M10 DIGRT*
   M11 CIX
   M12 INBUFF
   M13 ZTEMP1*
   M14 ZTEMP4*
   M15 ZTEMP3*
   M16 FLPTR
   M17 FPTR2*
   M18 LBPR1*
   M19 LBPR2*
   M20 LBUFF
   M21 PLYARG*
   M22 FPSCR/FSCR*
   M23 FPSCR1/FSCR1*
   M24 DEGFLG/RADFLG*

N. Power-Up and System Reset
   RAM sizing
     N1 RAMLO*,TRAMSZ*
     N2 TSTDAT*

   Diskette/cassette-boot
     N3 DOSINI
     N4 CKEY*
     N5 CASSBT*
     N6 CASINI
     N7 BOOT?*
     N8 DFLAGS*
     N9 DBSECT*
     N10 BOOTAD*

   Environmental control
     N11 COLDST
     N12 DOSVEC

```
[S RESET]
   N13 WARMST


P. Interrupts
   P1 CRITIC
   P2 POKMSK

   System Timers

     Real-time clock
       P3 RTCLOK

     System Timer 1
       P4 CDTMV1
       P5 CDTMA1

     System Timer 2
       P6 CDTMV2
       P7 CDTMA2

     System Timers 3-5
       P8 CDTMV3,CDTMV4,CDTMV5
       P9 CDTMF3,CDTMF4,CDTMF5

   RAM-interrupt vectors

     NMI-interrupt vectors
       P10 VDSLST
       P11 VVBLKI
       P12 VVBLKD



     IRQ-interrupt vectors
       P13 VIMIRQ
       P14 VPRCED
       P15 VINTER
       P16 VBREAK
       P17 VKEYBD
       P18 VSERIN
       P19 VSEROR
       P20 VSEROC
       P21 VTIMR1,VTIMR2,VTIMR4

   Hardware register updates
     P22 SDMCTL
     P23 SDLSTL,SDLSTH
     P24 GPRIOR
     P25 CHACT
     P26 CHBAS
     P27 PCOLRx,COLORx
```

Internal working variable
P28 INTEMP*

R. User areas
R1 (unlabeled)
R2 USAREA

This appendix contains descriptions of many of the data base variables; descriptions are included for all of the user-accessible variables and for some of the "internal" variables as well. Those variables that are not considered to be normally of interest to any user are flagged with an asterisk (*) after their names; the other variables can be of interest to one or more of the following classes of users:

- o End user.
- o Game developer.
- o Applications programmer.
- o System utility writer.
- o Language processor developer.
- o Device Handler Writer.

Each variable is specified by its system equate file name followed by its address (in hex) and the number of bytes reserved in the data base (in decimal), in the following form:

<name> [<address>,<size>]

For example:

MEMLO [02E7,2]

Note that most word (2 byte) variables are ordered with the least significant byte at the lower address.

A. MEMORY CONFIGURATION

See Section 4 for a general discussion of memory dynamics and section 7 for details of system initialization.

A1 MEMLO [02E7,2] -- User-free memory low address

MEMLO contains the address of the first location in the free memory region. The value is established by the OS during power-up and system reset initialization and is never altered by the OS thereafter.

A2 MEMTOP [02E5,2] -- User-free memory high address

MEMTOP contains the address of the first non-useable memory location above the free memory region. The value is established by the OS during power-up and system reset initialization; and then is re-established whenever the display is opened, based upon the requirements of the selected graphics mode.

A3 APPMHI [000E,2] -- User-free memory screen lower limit

APPMHI is a user-controlled variable that contains the address within the free memory region below which the Display Handler cannot go in setting up a display screen. This variable is initialized to zero by the OS at power-up.

A4 RAMTOP* [006A,1] -- Display Handler top of RAM address (MSB)

RAMTOP permanently retains the RAM top address that was contained in TRAMSZ (as described in N1) for the Display Handler's use. The value is set up as part of Handler initialization.

A5 RAMSIZ [02E4,1] -- Top of RAM address (MSB only)

RAMSIZ permanently retains the RAM top address that was contained in TRAMSZ (as described in N1).

B. TEXT/GRAPHICS SCREEN

See Section 5 for a discussion of the text and graphics screens and their Handlers.

Cursor Control

For the text screen and split-screen text window there is a visible cursor on the screen which shows the position of the next input or output operation. The cursor is represented by inversing the video of the character upon which it resides; but the cursor can be made invisible, at the user's option. The graphics screen always has an invisible cursor.

The cursor position is sensed by examining data base variables and can be moved by altering those same variables; in addition, when using the Screen Editor, there are cursor movement control codes that can be sent as data (as explained in Section 5).

B1 CRSINH [02F0,1] -- Cursor display inhibit flag

When CRSINH is zero, all outputs to the text screen will be followed by a visible cursor (inversed character); and when CRSINH is nonzero, no visible cursor will be generated.

CRSINH is set to zero by power-up, the [SYSTEM.RESET] or [BREAK] keys or an OPEN command to the Display Handler or Screen Editor.

Note that altering CRSINH does not cause the visible cursor to change states until the next output to the screen; if an immediate change to the cursor state is desired, without altering the screen data, follow the CRSINH change with the output of CURSOR UP, CURSOR DOWN, or some other innocuous sequence.

B2 ROWCRS [0054,1] and COLCRS [0055,2] -- Current cursor position

ROWCRS and COLCRS define the cursor location (row and column, respectively) for the next data element to be read from or written to the main screen segment. When in split-screen mode, the variables TXTROW and TXTCOL define the cursor for the text window at the bottom of the screen as explained in B4 below.

The row and column numbering start with the value zero, and increase in increments of one to the number of rows or columns minus 1; with the upper left corner of the screen being the origin (0,0).

ROWCRS is a single-byte variable with a maximum allowable value of 191 (screen modes 8-11); COLCRS is a 2-byte variable with a maximum allowable value of 319 (screen mode 8).

B3 OLDROW [005A,1] and OLDCOL [005B,2] -- Prior cursor position

OLDROW and OLDCOL are updated from ROWCRS and COLCRS before every operation. The variables are used only for the DRAW and FILL operations.

B4 TXTROW [0290,1] and TXTCOL [0291,2] -- Split-screen text cursor position

TXTROW and TXTCOL define the cursor location (row and column, respectively) for the next data element to be read from or written to the split-screen text window.

The row and column numbering start with the value zero, and increase in increments of one to 3 and 39, respectively; with the upper left corner of the split-screen text window being the origin (0,0).

Screen Margins

The text screen and split-screen text window have user-alterable left and right margins that define the normal domain of the text cursor.

B5 LMARGN [0052,1] -- Text column left margin

LMARGN contains the column number (0-39) of the text screen left margin; the text cursor will remain on or to the right of the left margin as a result of all operations, unless the cursor column variable is directly updated by the user (see B2 and B4 above). The default value for LMARGN is 2 and is established upon power-up or system reset.

B6 RMARGN [0053,1] -- Text column right margin

RMARGN contains the column number (0-39) of the text screen right margin; the text cursor will remain on or to the left of the right margin as a result of all operations, unless the cursor column variable is directly updated by the user (see B2 and B4 above). The default value for RMARGN is 39 and is established upon power-up or system reset.

Color Control

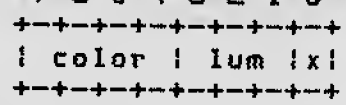
As part of the stage 2 VBLANK process (see Section 6), the values of nine data base variables are stored in corresponding hardware color control registers. The color registers are divided into two groups: the player/missile colors and the playfield colors. The playfield color registers are utilized by the different screen modes as shown in Appendix H. The player/missile color registers are not used by the standard OS.

B7 PCOLR0 - PCOLR3 [02C0,4] -- Player/missile graphics colors

Each color variable is stored in the corresponding hardware register as shown below:

PCOLR0 [02C0]	COLPM0 [D012]
PCOLR1 [02C1]	COLPM1 [D013]
PCOLR2 [02C2]	COLPM2 [D014]
PCOLR3 [02C3]	COLPM3 [D015]

Each color variable has the format shown below:



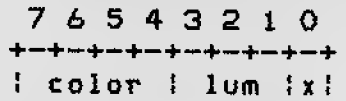
See Appendix H for information regarding the color and luminance field values.

B8 COLOR0 - COLOR4 [02C5,5] -- Playfield colors

Each color variable is stored in the corresponding hardware register as shown below:

COLOR0 [02C4]	COLPF0 [D016]
COLOR1 [02C5]	COLPF1 [D017]
COLOR2 [02C6]	COLPF2 [D018]
COLOR3 [02C7]	COLPF3 [D019]
COLOR4 [02C8]	COLBK [D01A]

Each color variable has the format shown below:



See Appendix H for information regarding the color and luminance field values.

Text Scrolling

The text screen or split-screen text window "scrolls" upward whenever one of the two conditions shown below occurs:

- A text line at the bottom row of the screen extends past the right margin.
- A text line at the bottom row of the screen is terminated by an EOL.

Scrolling has the effect of removing the entire logical line that starts at the top of the screen and then moving all subsequent lines upward to fill in the void. The cursor will also move upward if the logical line deleted exceeds one physical line.

B9 SCRFLG* [02BB,1] -- Scroll flag

SCRFLG is a working variable that counts the number of physical lines minus 1 that were deleted from the top of the screen; since a logical line ranges in size from 1 to 3, SCRFLG ranges from 0 to 2.

Attract Mode

Attract mode is a mechanism that protects the television screen from having patterns "burned into" the phosphors due to a fixed display being left on the screen for extended periods of time. When the computer is left unattended for more than 9 minutes, the color intensities are limited to 50 percent of maximum and the hues are continually varied every 8.3 seconds. Pressing any keyboard data key will be sufficient to remove the attract mode for 9 more minutes.

As part of the stage 2 VBLANK process, the color registers from the data base are sent to the corresponding hardware color registers; before they are sent, they undergo the following transformation:

    hardware register = database variable XOR COLRSH AND DRKMSK

Normally COLRSH = $00 and DRKMSK = $FE, thus making the above calculation a null operation; however, once attract mode becomes active, COLRSH = the content of RTCLOK+1 and DRKMSK = $F6, that has the effect of modifying all of the colors and keeping their luminance always below the 50 percent level.

Since RTCLOK+1 is incremented every 256/60 of a second and since the least significant bit of COLRSH is of no consequence, a

color/lum change will be effected every 8.3 seconds (512/60).

B10 ATRACT [004D,1] -- Attract mode timer and flag

ATRACT is the timer (and flag) that controls the initiation and termination of attract mode. Whenever a keyboard key is pressed, the keyboard IRQ service routine sets ATRACT to zero, thus terminating attract mode; the [BREAK] key logic behaves accordingly. As part of the stage 1 VBLANK process, ATRACT is incremented every 4 seconds; if the value exceeds 127 (after 9 minutes without keyboard activity), the value of ATRACT will be set to $FE and will retain that value until attract mode is terminated.

Since the attract mode is prevented and terminated by the OS based only upon keyboard activity, some users can want to reset ATRACT based upon Atari-controller event detection, user-controlled Serial I/O bus activity or any other signs of life.

B11 COLRSH* [004F,1] -- Color shift mask

COLRSH has the value $00 when attract mode is inactive, thus effecting no change to the screen colors; when attract mode is active, COLRSH contains the current value of the timer variable middle digit (RTCLOK+1).

B12 DRKMSK* [004E,1] -- Dark (luminance) mask

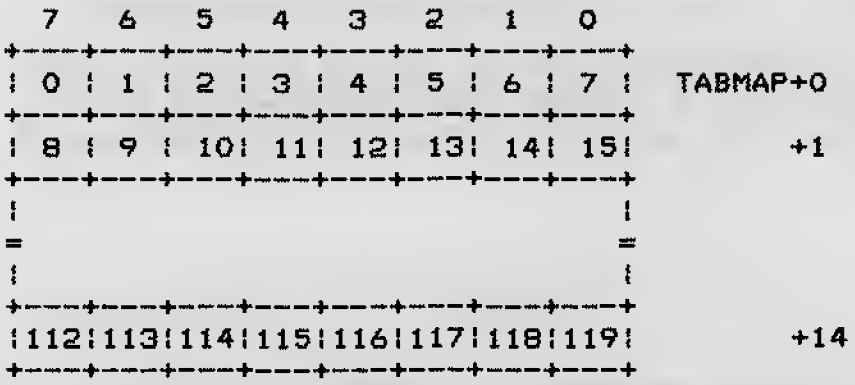
DRKMSK has the value $FE when attract mode is inactive, which does not alter the luminance; and has the value $F6 when attract mode is active, which forces the most significant bit of the luminance field to zero, thus guaranteeing that the luminance will never exceed 50 percent.

Tabbing

See Section 5 for a discussion of the use of tabs in conjunction with the Screen Editor.

B13 TABMAP [02A3,15] -- Tab stop setting map

The tab settings are retained in a 15-byte (120 bit) map, where a bit value of 1 indicates a tab setting; the diagram below shows the mapping of the individual bits to tab positions.



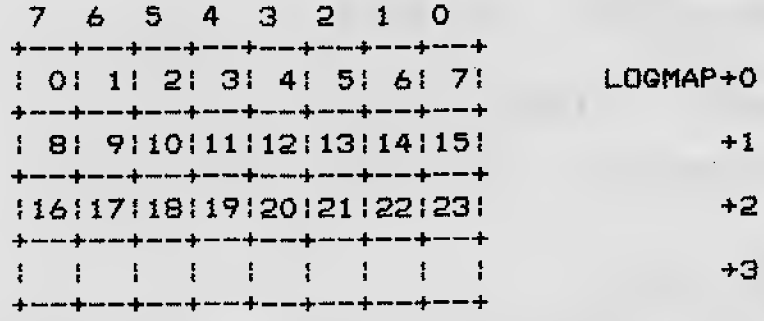
Whenever the Display Handler or Screen Editor is opened, this map is initialized to contain the value of $01 in every byte, thus providing the default tab stops at 7, 15, 23, etc.

Logical Text Lines

The text screen is invisibly divided into logical lines of text, each comprising from one to three physical lines of text. The screen is initialized to 24 logical lines of one physical line each; but data entry and/or data insertion can increase the size of a logical line to two or three physical lines.

B14 LOGMAP* [02B2,4] -- Logical line starting row map

The beginning physical line number for each logical line on the screen is retained in a four byte (32 bit) map, where a bit value of one indicates the start of a logical line; the diagram below shows the mapping of the individual bits to physical line (row) numbers.



The map bits are all set to 1 whenever the text screen is opened or cleared. From that point, the map is updated as logical lines are entered, edited and deleted from the screen.

B15 LOGCOL* [0063,1] -- Cursor/logical line column number

LOGCOL contains the logical-line column number for the current cursor position; note that a logical line can comprise up to three physical lines. This variable is for the internal use of the Display Handler.

Split Screen

The Display Handler and Screen Editor together support the operation of a split-screen mode (see Section 5) in which the main portion of the screen is in one of the graphics modes and is controlled by the Display Handler, and there are 4 physical lines in the text window at the bottom of the screen which is controlled by the Screen Editor.

B16 BOTSCR* [02BF,1] -- Text screen lines count

BOTSCR contains the number of lines of text for the current screen: 24 for mode 0 or 4 for a split-screen mode. The Handler also uses this variable as an indication of the split-screen status; tests are made for the specific values 4 and 24.

DRAW/FILL Function

The DRAW function line drawing algorithm is shown below translated to the PASCAL language from assembly language.

```
    NEWROW := ROWCRS; NEWCOL := COLCRS;

    DELTAR := ABS (NEWROW-OLDROW);
    ROWINC := SIGN (NEWROW-OLDROW); { +1 or -1 }

    DELTAC := ABS (NEWCOL-OLDCOL);
    COLINC := SIGN (NEWCOL-OLDCOL); { +1 or -1 }

    ROWAC := 0; COLAC := 0;
    ROWCRS := OLDROW; COLCRS := OLDCOL;

    COUNTR := MAX (DELTAC,DELTAR);
    ENDPT := COUNTR;
    IF COUNTR = DELTAC
       THEN ROWAC := ENDPT DIV 2
       ELSE COLAC := ENDPT DIV 2;

    WHILE COUNTR > 0 DO
       BEGIN
```

```
            ROWAC := ROWAC + DELTAR;
            IF ROWAC >= ENDPT
               THEN
                  BEGIN
                     ROWAC := ROWAC - ENDPT;
                     ROWCRS := ROWCRS + ROWINC
                  END;

            COLAC := COLAC + DELTAC;
            IF COLAC >= ENDPT
               THEN
                  BEGIN
                     COLAC := COLAC - ENDPT;
                     COLCRS := COLCRS + COLINC
                  END;

            PLOT_POINT; { point defined by ROWCRS and COLCRS }

            IF FILFLG <> 0 THEN FILL_LINE;

            COUNTR := COUNTR - 1

         END;
```

The FILL function algorithm (FILL_LINE above) is described briefly in Section 5.

B17 FILDAT [02FD,1] -- Fill data

FILLDAT contains the fill region data value as part of the calling sequence for a FILL command as described in Section 5.

B18 FILFLG* [02B7,1] -- Fill flag

FILFLG indicates to the shared code within the Display Handler whether the current operation is FILL (FILFLG <> 0) or DRAW (FILFLG = 0).

B19 NEWROW* [0060,1] and NEWCOL* [0061,2] -- Destination point

NEWROW and NEWCOL are initialized to the values in ROWCRS and COLCRS, which represent the destination endpoint of the DRAW/FILL command. This is done so that ROWCRS and COLCRS can be altered during the performance of the command.

B20 HOLD4* [02BC,1] -- Temporary storage

HOLD4 is used to save and restore the value in ATACHR during the FILL process; ATACHR is temporarily set to the value in FILDAT to accomplish the filling portion of the command.

B21 ROWINC* [0079,1] and COLINC* [007A,1] -- Row/column increment/decrement

ROWINC and COLINC are the row and column increment values; they are each set to +1 or -1 to control the basic direction of line drawing. ROWINC and COLINC represent the signs of NEWROW - ROWCRS and NEWCOL - COLCRS, respectively.

B22 DELTAR* [0076,1] and DELTAC* [0077,2] -- Delta row and delta column

DELTAR and DELTAC contain the absolute values of NEWROW - ROWCRS and NEWCOL - COLCRS, respectively; together with ROWINC and COLINC, they define the slope of the line to be drawn.

B23 COUNTR* [007E,2] -- Draw iteration count

COUNTR initially contains the larger of DELTAR and DELTAC, that is the number of iterations required to generate the desired line. COUNTR is then decremented after every point on the line is plotted, until it reaches a value of zero.

B24 ROWAC* [0070,2] and COLAC* [0072,2] -- Accumulators

ROWAC and COLAC are working accumulators that control the row-and column-point plotting and increment (or decrement) function.

B25 ENDPT* [0074,2] -- Line length

ENDPT contains the larger of DELTAR and DELTAC, and is used in conjunction with ROWAC/COLAC and DELTAR/DELTAC to control the plotting of line points.

Displaying Control Characters

Often it is useful to have ATASCII control codes (such as CLEAR, CURSOR UP, etc). displayed in their graphic forms instead of having them perform their control function. This display capability is provided in two forms when outputting to the Screen Editor: 1) a data content form in which a special character (ESC) precedes each control character to be displayed and 2) a mode control form.

Escape (Display Following Control Character)

Whenever an ESC character is detected by the Screen Editor, the next character following this code is displayed as data, even if it would normally be treated as a control code; the EOL code is the sole exception. It is always treated as a control code. The sequence ESC ESC will cause the second ESC character to be displayed.

B26 ESCFLG* [02A2,1] -- Escape flag

ESCFLG is used by the Screen Editor to control the escape sequence function; the flag is set (to $80) by the detection of an ESC character ($1B) in the data stream and is reset (to 0) following the output of the next character.

Display Control Characters Mode

When it is desired to display ATASCII control codes other than EOL in their graphics form, but not have an ESC character associated with each control code, a display mode can be established by setting a flag in the data base. This capability is used by language processors when displaying high-level language statements, that can contain control codes as data elements.

B27 DSPFLG [02FE,1] -- Display control characters flag

When DSPFLG is nonzero, ATASCII control codes other than EOL are treated as data and displayed on the screen when output to the Screen Editor. When DSPFLG is zero, ATASCII control codes are processed normally.

DSPFLG is set to zero by Power-up and [SYSTEM.RESET].

Bit-Mapped Graphics

A number of temporary variables are used by the Display Handler when handling data elements (pixels) going to or from the screen; of interest here are those variables that are used to control the packing and unpacking of graphics data, where a memory byte typically contains more than one data element (for example, screen mode 8 contains 8 pixels per memory byte).

B28 DMASK* [02A0,1] -- Pixel location mask

DMASK is a mask that contains zeros for all bits that do not correspond to the specific pixel to be operated upon, and 1's for all bits that do correspond. DMASK can contain the values shown below in binary notation:

```
    11111111  --  screen modes 1 and 2; one pixel per byte.

    11110000  --  screen modes 9-11; two pixels per byte.
    00001111

    11000000  --  screen modes 3, 5 and 7; four pixels per byte.
    00110000
    00001100
    00000011

    10000000  --  screen modes 4, 6 and 8; eight pixels per byte.
    01000000

    00000010
    00000001
```

B29 SHFAMT* [006F,1] -- Pixel justification

SHFAMT indicates the amount to shift the right-justified pixel data on output, or the amount to shift the input data to right justify it on input. The value is always the same as for DMASK prior to the justification process.

Internal Working Variables

B30 HOLD1* [0051,1] -- Temporary storage

B31 HOLD2* [029F,1] -- Temporary storage

B32 HOLD3* [029D,1] -- Temporary storage

B33 TMPCHR* [0050,1] -- Temporary storage

B34 DSTAT* [004C,1] -- Display status

B35 DINDEX [0057,1] -- Display mode

DINDEX contains the current screen mode obtained from the low order four bits of the most recent OPEN AUX1 byte.

B36 SAVMSC [0058,2] -- Screen Memory Address

SAVMSC contains the lowest address of the screen data region; the data at that address is displayed at the upper left corner of the screen.

B37 OLDCHR* [005D,1] -- Cursor character save/restore

OLDCHR retains the value of the character under the visible text cursor; this variable is used to restore the original character value when the cursor is moved.

B38 OLDADR* [005E,2] -- Cursor memory address

OLDADR retains the memory address of the current visible text cursor location; this variable is used in conjunction with OLDCHR (B37) to restore the original character value when the cursor is moved.

B39 ADRESS* [0064,2] -- Temporary storage

B40 MLTTMP/OPNTMP/TOADR* [0066,2] -- Temporary storage

B41 SAVADR/FRMADR* [0068,2] -- Temporary storage

B42 BUFCNT* [006B,1] -- Screen Editor current logical line size

B43 BUFSTR* [006C,2] -- Temporary storage

B44 SWPFLG* [007B,1] -- Split-screen cursor control

In split-screen mode, the graphics cursor data and the text window cursor data are frequently swapped as shown below in order to get the variables associated with the region being accessed into the ROWCRS-OLDADR variables.

```
ROWCRS B2 -------- TXTROW B4
COLCRS B2 -------- TXTCOL B4
DINDEX B35 ------- TINDEX B49
SAVMSC B36 ------- TXTMSC B52
OLDROW B3 -------- TXTOLD B53
OLDCOL B3 --------    "     "
OLDCHR B37 -------    "     "
OLDADR B38 -------    "     "
```

SWPFLG is used to keep track of what data set is currently in the ROWCRS-OLDADR region; SWPFLG is equal to $FF when split-screen text window cursor data is in the main region, otherwise SWPFLG is equal to 0.

B45 INSDAT* [007D,1] -- Temporary storage

B46 TMPROW* [02B8,1] and TMPCOL* [02B9,2] -- Temporary storage

B47 TMPLBT* [02A1,1] -- Temporary storage

B48 SUBTMP* [029E,1] -- Temporary storage

B49 TINDEX* [0293,1] -- Split screen text window screen mode

TINDEX is the split-screen text window equivalent of DINDEX and is always equal to zero when SWPFLG is equal to zero (see B44).

B50 BITMSK* [006E,1] -- Temporary storage

B51 LINBUF* [0247,40] -- Physical line buffer

LINBUF is used to temporarily buffer one physical line of text when the Screen Editor is moving screen data.

B52 TXTMSC [0294,2] -- Split screen memory address

TXTMSC is the split-screen text window version of SAVMSC (B36).

See B44 for more information.

B53 TXTOLD* [0296,6] -- Split screen cursor data

See B44 for more information.

Internal Character Code Conversion

Two variables are used to retain the current character being processed (for both reading and writing); ATACHR contains the value passed to or from CIO, and CHAR contains the internal code corresponding to the value in ATACHR. Because the hardware does not interpret ATASCII characters directly, the transformations shown below are applied to all text data read and written:

ATASCII CODE	INTERNAL CODE
00-1F	40-5F
20-3F	00-1F
40-5F	20-3F
60-7F	60-7F
80-9F	C0-DF

A0-BF	B0-9F
C0-DF	A0-BF
E0-FF	E0-FF

See P26 for more information.

B54 ATACHR [02FB,1] -- Last ATASCII character or plot point

ATACHR contains the ATASCII value for the most recent character read or written, or the value of the graphics point. This variable can also be considered to be a parameter of the FILL/DRAW commands, as the value in ATACHR will determine the line color when a DRAW or FILL is performed.

B55 CHAR* [02FA,1] -- Internal character code

CHAR contains the internal code value for the most recent character read or written.

C. DISKETTE HANDLER

See Section 5 for a discussion of the resident Diskette Handler.

C1 BUFADR* [0015,2] -- Data buffer pointer

BUFADR acts as temporary page zero pointer to the current diskette buffer.

C2 DSKTIM* [0246,1] -- Disk format operation timeout time

DSKTIM contains the timeout value for SIO calling sequence variable DTIMLO (see Section 9). DSKTIM is set to 160 (which represents a 171-second timeout) at initialization time, and is updated after each diskette status request operation. It contains the value returned in the third byte of the status frame (see Section 5). Note that all diskette operations other than format have a fixed (7) second timeout, established by the Diskette Handler.

D. CASSETTE

See Section 5 for a general description of the Cassette Handler. The cassette uses the Serial I/O bus hardware, but does not conform with the Serial I/O bus protocol as defined in Section 9. Hence, the Serial

I/O utility (SIO) has cassette specific code within it. Some variables in this subsection are utilized by SIO and some by the Cassette Handler.

Baud Rate Determination

The input baud rate is assumed to be a nominal 600 baud, but will be adjusted, if necessary, by the SIO routine to account for drive-motor variations, stretched tape, etc. The beginning of every cassette record contains a pattern of alternating 1's and zeros that is used solely for speed correction; by measuring the time to read a fixed number of bits, the true-receive baud rate is determined and the hardware adjusted accordingly. Input baud rates ranging from 318 to 1407 baud can theoretically be handled using this technique.

The input baud rate is adjusted by setting the POKEY counter that controls the bit sampling period.

D1 CBAUDL* [02EE,1] and CBAUDH* [02EF,1] -- Cassette baud rate

Initialized to 05CC hex, which represents a nominal 600 baud. After baud rate calculation, these variables will contain POKEY counter values for the corrected baud rate.

D2 TIMFLG* [0317,1] -- Baud rate determination timeout flag

TIMFLG is used by SIO to timeout an unsuccessful baud rate determination. The flag is initially set to 1, and if it attains a value of zero (after 2 seconds) before the first byte of the cassette record has been read, the operation will be aborted. See also H24.

D3 TIMER1* [030C,2] and TIMER2* [0310,2] -- Baud rate timers

These timers contain reference times for the beginning and end of the fixed bit pattern receive period. The first byte of each timer contains the then current vertical line counter value read from ANTIC, and the second byte of each timer contains the then current value of the least significant byte of the OS real time clock (RTCLOK+2).

The difference between the timers is converted to raster pair counts and is then used to perform a table lookup with interpolation to determine the new values for CBAUDL and CBAUDH.

D4 ADDCOR* [030E,1] -- Interpolation adjustment variable

ADDCOR is a temporary variable used for the interpolation calculation of the above computation.

D5 TEMP1* [0312,2] -- Temporary storage

D6 TEMP3* [0315,1] -- Temporary storage

D7 SAVID* [0316,1] -- Serial in data detect

SAVID is used to retain the state of SKSTAT [D20F] bit 4 (serial data in); it is used to detect (and is updated after) every bit arrival.

Cassette Mode

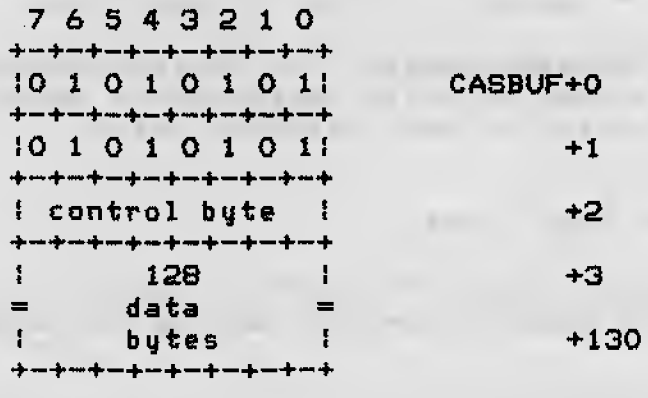
D8 CASFLG* [030F,1] -- Cassette I/O flag

CASFLG is used internally by SIO to control the program flow through shared code. A value of zero indicates that the current operation is a standard Serial I/O bus operation, and a nonzero value indicates a cassette operation.

Cassette Buffer

D9 CASBUF* [03FD,131] -- Cassette record buffer

CASBUF is the buffer used by the Cassette Handler for the packing and unpacking of cassette-record data, and by the initialization cassette-boot logic. The format for the standard cassette record in the buffer is shown below:



See Section 5 for an explanation of the standard cassette-record format.

D10 BLIM* [028A,1] -- Cassette record data size

BLIM contains the count of the number of data bytes in the current cassette record being read. BLIM will have a value ranging from 1 to 128, depending upon the record control byte as explained in Section 5.

D11 BPTR* [003D,1] -- Cassette-record data index

BPTR contains an index into the data portion of the cassette record being read or written. The value will range from 0 to the then current value of BLIM. When BPTR equals BLIM then the buffer (CASBUF) is full if writing or empty if reading.

Internal Working Variables

D12 FEOF* [003F,1] -- Cassette end-of-file flag

FEOF is used by the Cassette Handler to flag the detection of an end of file condition (control byte = $FE). FEOF equal to zero indicates that an EOF has not yet been detected, and a nonzero value indicates that an EOF has been detected. The flag is reset at every OPEN.

D13 FTYPE* [003E,1] -- Interrecord gap type

FTYPE is a copy of ICAX2Z from the OPEN command and indicates the type of interrecord gap selected; a positive value indicates normal record gaps, and a negative value indicates continuous mode gaps.

D14 WMODE* [0289,1] -- Cassette read/write mode flag

WMODE is used by the Cassette Handler to indicate whether the current operation is a read or write operation; a value of zero indicates read, and a value of $80 indicates write.

D15 FREQ* [0040,1] -- Beep count

FREQ is used to retain and count the number of beeps requested of the BEEP routine by the Cassette Handler during the OPEN command process.

E. KEYBOARD

See Section 5 for a general description of the Keyboard Handler.

Key Reading and Debouncing

The console key code register is read in response to an IRQ interrupt that is generated whenever a key stroke is detected by the hardware. The key code is compared with the prior key code accepted (CH1); if the codes are not identical, then the new code is accepted and stored in the key code FIFO (CH) and in the prior key code variable (CH1). If the codes are identical, then the new code is accepted only if a suitable key debounce delay has transpired since the prior value was accepted.

If the key code read and accepted is the code for [CTRL] 1, then the display start/stop flag (SSFLAG) is complemented and the value is not stored in the key code FIFO (CH).

In addition to the reading of the key data, SRTIMR is set to $30 for all interrupts received (see E8), and ATRACT is set to 0 whenever a new code is accepted (see B10).

The Keyboard Handler obtains all key data from CH; whenever a code is extracted from that 1-byte FIFO, the Handler stores a value of $FF to the FIFO to indicate that the code has been read. See Section 5 for further discussion of the Keyboard Handler's processing of the key codes.

E1 CH1* [02F2,1] -- Prior keyboard character code.

CH1 contains the key code value of the key most recently read and accepted.

E2 KEYDEL* [02F1,1] -- Debounce delay timer.

KEYDEL is set to a value of 3 whenever a key code is accepted, and is decremented every 60th of a second by the stage 2 VBLANK process (until it reaches zero).

E3 CH [02FC,1] -- Keyboard character code FIFO.

CH is a 1-byte FIFO that contains either the value of the most recently read and accepted key code or the value $FF (which indicates that the FIFO is empty). The FIFO is normally read by the Keyboard Handler, but can be read by a user program.

Key data can also be stored into CH by the Autorepeat logic as explained in the discussion relating to E8.

Start/Stop

Display Handler and Screen Editor output to the text or graphics mode screen can be stopped and started (without losing any of the output data) through the use of the [CTRL] 1 key combination. Each key depression toggles a flag that is monitored by the above mentioned Handlers. When the flag is nonzero, the handlers wait for it to go to zero before continuing any output.

E4 SSFLAG [02FF,1] -- Start/stop flag

The flag is normally zero, indicating that screen output is not to be stopped. The flag is complemented by every occurrence of the [CTRL] 1 key combination by the keyboard IRQ service routine.

The flag is set to zero upon power-up, [SYSTEM.RESET] or [BREAK] key processing.

[BREAK] Key

E5 BRKKEY [0011,1] -- [BREAK] key flag

BRKKEY is used to indicate that the [BREAK] key has been pressed. The value is normally nonzero and is set to zero whenever the [BREAK] key is pressed. The code that detects and processes the [BREAK] condition (flag = 0) should set the flag nonzero again.

BRKKEY is monitored by the following OS routines: Keyboard Handler, Display Handler, Screen Editor, Cassette Handler, xx? The detection of a [BREAK] condition during an I/O operation will cause the operation to be aborted and a status of $80 to be returned to the user.

The flag is set to nonzero upon Power-up, [SYSTEM.RESET] or upon aborting a pending I/O operation.

[SHIFT]/[CONTROL] Lock

The keyboard control has three different modes for code generation that apply to the alphabetic keys A through Z: 1) normal, 2) caps lock, and 3) control lock.

In normal mode, all unmodified alphabetic character keys generate the lowercase letter ATASCII code ($61-7A).

In caps lock mode, all unmodified alphabetic character keys generate the uppercase letter ATASCII code ($41-5A).

In control lock mode, all unmodified alphabetic character keys generate the control letter ATASCII code ($01-1A).

In all three modes, any alphabetic character key that is modified (by being pressed in conjunction with the [SHIFT] or [CTRL] key) will generate the desired modified code.

E6 SHFLOK [02BE,1] -- Shift/control lock control flag

SHFLOK normally has one of three values:

$00 = normal mode (no locks in effect).
$40 = caps lock.
$80 = control lock.

SHFLOK is set to $40 upon Power-up and [SYSTEM.RESET] and is modified thereafter by the OS only when the [CAPS.LOWER] key is pressed (either by itself or in conjunction with the [SHIFT] or [CTRL] key).

E7 HOLDCH* [007C,1] -- Character holding variable

HOLDCH is used to retain the current character value prior to the [SHIFT]/[CONTROL] logic process.

Autorepeat

The Autorepeat feature responds to the continuous depression of a keyboard key by replicating the key code 10 times per second, after an initial 1/2 second delay. The timer variable SRTIMR is used to control both the initial delay and the repeat rate.

Whenever SRTIMR is equal to zero and a key is being held down, the value of the key code is stored in the key code FIFO (CH). This logic is part of the stage 2 VBLANK process.

E8 SRTIMR* [022B,1] -- Autorepeat timer

SRTIMR is controlled by two independent processes: 1) the keyboard IRQ service routine, which establishes the initial delay value and 2) the stage 2 VBLANK routine that establishes the repeat rate, decrements the timer and implements the auto repeat logic.

Inverse Video Control

The Keyboard Handler allows the direct generation of more than half of the 256 ATASCII codes; but codes $80-9A and codes $A0- FC can be generated only with the "inverse video mode" active. The ATARI key acts as an on/off toggle for this mode, and all characters (except for screen editing control characters) will be subject to inversion when the mode is active.

E9 INVFLG [02B6,1] -- Inverse video flag

INVFLG is normally zero, indicating that normal video ATASCII codes (bit 7 = 0) are to be generated from keystrokes; whenever INVFLG is nonzero, inverse video ATASCII codes (bit 7 = 1) will be generated. The special control codes are exempt from this bit manipulation.

INVFLG is set to zero by power-up and system reset.

The Keyboard Handler inverts bit 7 of INVFLG whenever the ATARI key is pressed; the lower order bits are not altered and are assumed to be zero.

The Keyboard Handler's "exclusive or's" (XOR's) the ATASCII key data with the value in INVFLG at all times; the normal values of $00 and $80 thus lead to control of the inverse video bit (bit 7).

Console Keys: [SELECT], [START], and [OPTION]

The console keys are sensed directly from the hardware register CONSOL [D01F]; see the ATARI Home Computer Hardware Manual for details.

F. PRINTER

See Section 5 for a general description of the Printer Handler.

Printer-Buffer

F1 PRNBUF* [03C0,40] -- Printer-record buffer

PRNBUF is the buffer used by the Printer Handler for packing printer data to be sent to the device controller. The buffer is 40 bytes long

and contains nothing but printer data.

F2 PBUFSZ* [001E,1] -- Printer-record size

PBUFSZ contains the size of the Printer-record for the current mode selected; the modes and respective sizes (in decimal bytes) are shown below:

Normal	40	
Double width	20	(not currently supported by the device)
Sideways	29	

Status request 4

F3 PBPNT* [001D,1] -- Printer-buffer index

PBPNT contains the current index to the Printer-buffer. PBPNT ranges in value from zero to the value of PBUFSZ.

Internal Working Variables

F4 PTEMP* [001F,1] -- Printer Handler temporary data save

PTEMP is used by the Printer Handler to temporarily save the value of a character to be output to the printer.

F5 PTIMOT* [001C,1] -- Printer timeout value

PTIMOT contains the timeout value for SIO calling sequence variable DTIMLO (see Section 9); PTIMOT is set to 30 (which represents a 32 second timeout) at intialization time, and is updated after each printer status request operation to contain the value returned in the third byte of the status frame (see Section 5).

G. CENTRAL I/O ROUTINE (CIO)

See Section 5 for a description of the Central I/O Utility.

User Call Parameters

CIO call parameters are passed primarily through an I/O Control Block (IOCB); although additional device status information can be returned in DVSTAT, and Handler information is obtained from the device table (HATABS).

I/O Control Block

IOCB is the name applied collectively to the 16 bytes associated with each of the 8 provided control structures; see Section 5.

G1 IOCB [0340,16] -- I/O Control Block

The label IOCB is the location of the first byte of the first IOCB in the data base. For VIDs G2 through G10, the addresses given are for IOCB #0 only, the addresses for all of the IOCB's are shown below:

0340-034F	IOCB #0
0350-035F	IOCB #1
0360-036F	IOCB #2
0370-037F	IOCB #3
0380-038F	IOCB #4
0390-039F	IOCB #5
03A0-03AF	IOCB #6
03B0-03BF	IOCB #7

G2 ICHID [0340,1] -- Handler ID

See Section 5. Initialized to $FF at power-up and system reset.

G3 ICDNO [0341,1] -- Device number

See Section 5.

G4 ICCOM [0342,1] -- Command byte

See Section 5.

G5 ICSTA [0343,1] -- Status

See Section 5.

G6 ICBAL, ICBAH [0344,2] -- Buffer address

See Section 5.

G7 ICPTL,ICPTH [0346,2] -- PUT BYTE vector

See Section 5. Initialized to point to CIO's "IOCB not OPEN" routine at power-up and system reset.

G8 ICBLL,ICBLH [0348,2] -- Buffer length / byte count

See Section 5.

G9 ICAX1,ICAX2 [034A,2] -- Auxiliary information

See Section 5.

G10 ICSPR [034C,4] -- Spare bytes for Handler use

There is no fixed assignment of these four bytes; the Handler associated with an IOCB can or may not use these bytes.

Device Status

G11 DVSTAT [02EA,4] -- Device status

See Section 5 for a discussion of the GET STATUS command.

Device Table

G12 HATABS [031A,3B] -- Device table

See Section 9 for a description of the device table.

CIO/Handler Interface Parameters

Communication between CIO and a Handler is accomplished using the 6502 machine registers, and a data structure called the Zero-page IOCB (ZIOCB). The ZIOCB is essentially a copy of the particular IOCB being used for the current operation.

G13 ZIOCB (IOCBAS) [0020,16] -- Zero-page IOCB

The Zero-page IOCB is an exact copy (except as noted in the discussions that follow) of the IOCB specified by the 6502 X register upon entry to CIO; CIO copies the outer level IOCB to the Zero-page IOCB, performs the indicated function, moves the (possibly altered) Zero-page IOCB back to the outer level IOCB, and then returns to the caller.

Although both the outer level IOCB and the Zero-page IOCB are defined to be 16 bytes in size, only the first 12 bytes are moved by CIO.

G14 ICHIDZ [0020,1] -- Handler index number

See Section 5. Set to $FF on CLOSE.

G15 ICDNOZ [0021,1] -- Device drive number

See Section 5.

G16 ICCOMZ [0022,1] -- Command byte

See Section 5.

G17 ICSTAZ [0023,1] -- Status byte

See Section 5.

G18 ICBALZ,ICBALH [0024,2] -- Buffer address

See Section 5. This pointer variable is modified by CIO in the course of processing some commands; however, the original value is restored before returning to the caller.

G19 ICPTLZ,ICPTHZ

See Section 5. Set to point to CIO's "IOCB not OPEN" routine on CLOSE.

G20 ICBLLZ,ICBLHZ [0028,2] -- Buffer length / byte count

See Section 5. This double-byte variable, which starts out representing the buffer length, is modified by CIO in the course

of processing some commands; then, before returning to the caller, the transaction byte count is stored therein.

G21 ICAX1Z,ICAX2Z [002A,2] -- Auxiliary information

See Section 5.

G22 ICSPRZ (ICIDNO,CIOCHR) [002C,4] -- CIO working variables

ICSPRZ and ICSPRZ+1 are used by CIO in obtaining the appropriate Handler entry point from the handler's vector table (see Section 9).

ICSPRZ+2 is also labeled ICIDNO and retains the value of the 6502 X register from CIO entry. The X register is loaded from ICIDNO as CIO returns to the caller.

ICSPRZ+3 is also labeled CIOCHR and retains the value of the 6502 A register from CIO entry, except for data reading type commands, in which case the most recent data byte read is stored in CIOCHR. The 6502 A register is loaded from CIOCHR as CIO returns to the caller.

Internal Working Variables

G23 ICCOMT* [0017,1] -- Command table index

ICCOMT is used as an index to CIO's internal command table, which maps command byte values to Handler entry offsets (see Section 9 for more information). ICCOMT contains the value from ICCOMZ except when ICCOMZ is greater than $0E, in which case ICCOMT is set to $0E.

G24 ICIDNO* [002E,1] -- CIO call X register save/restore

See G22.

G25 CIOCHR* [002F,1] -- CIO call A register save/restore

See G22.

H. SERIAL I/O ROUTINE (SIO)

See Section 9 for discussions relating to SIO.

User Call Parameters

SIO call parameters are passed primarily through a Device Control Block; although an additional "noisy bus" option exists that is selectable through a separate variable.

Device Control Block

H1 DCB [0300,12] -- Device Control Block

DCB is the name applied collectively to the 12 bytes at locations 0300-030B. These bytes provide the parameter passing mechanism for SIO and are described individually below.

H2 DDEVIC [0300,1] -- Device bus ID

See Section 9.

H3 DUNIT [0301,1] -- Device unit number

See Section 9.

H4 DCOMND [0302,1] -- Device command

See Section 9.

H5 DSTATS [0303,1] -- Device status

See Section 9.

H6 DBUFLO,DBUFHI [0304,2] -- Handler buffer address

See Section 9.

H7 DTIMLO [0306,1] -- Device timeout

See Section 9.

H8 DBYTLO,DBYTHI [0308,2] -- Buffer length / byte count

See Section 9.

H9 DAUX1,DAUX2 [030A,2] -- Auxiliary information

See Section 9.

Bus Sound Control

H10 SOUNDR [0041,1] -- Quiet/noisy I/O flag

SOUNDR is a flag used to indicate to SIO whether noise is to be generated on the television audio circuit when Serial I/O bus activity is in progress. SOUNDR equal to zero indicates that sound is to be inhibited, and nonzero indicates that sound is to be enabled. SIO sets SOUNDR to 3 at power-up and system reset.

Serial Bus Control

Retry Logic

SIO will attempt one complete command retry if the first attempt is not error free, where a complete command try consists of up to 14 attempts to send (and acknowledge) a command frame, followed by a single attempt to receive COMPLETE and possibly a data frame.

H11 CRETRY* [0036,1] -- Command frame retry counter

CRETRY controls the inner loop of the retry logic, that associated with sending and receiving an acknowledgement of the command frame. CRETRY is set to 13 by SIO at the beginning of every command initiation, thus allowing for an initial attempt and up to 13 additional retries.

H12 DRETRY* [0037,1] -- Device retry counter

DRETRY controls the outer loop of the retry logic, that associated with initiating a command retry after a failure subsequent to the command frame acknowledgement. DRETRY is set to 1 by SIO at entry, thus allowing for an initial attempt and 1 additional retry.

Checksum

The Serial I/O bus protocol specifies that all command and data frames must contain a checksum validation byte; this byte is the arithmetic sum (with end-around carry) of all of the other bytes in the frame.

H13 CHKSUM* [0031,1] -- Checksum value

CHKSUM contains the frame checksum as computed by SIO for all frame transfers.

H14 CHKSNT* [003B,1] -- Checksum sent flag

CHKSNT indicates to the serial bus transmit interrupt service routine whether the frame checksum byte has been sent yet. CHKSNT equal to zero indicates that the checksum byte has not yet been sent; after the checksum is sent, CHKSNT is then set nonzero.

H15 NOCKSM* [003C,1] -- No checksum follows data flag

NOCKSM is a flag used to communicate between the SIO top level code and the Serial bus receive interrupt service routine that the next input will not be followed by a checksum byte. A value of zero specifies that a checksum byte will follow, nonzero specifies that a checksum byte will not follow.

Data Buffering

General Buffer Control

H16 BUFRLO* [0032,1] and BUFRHI* [0033,1] -- Next byte address

BUFRLO and BUFRHI comprise a pointer to the next buffer location to be read from or written to. For a data frame transfer, the pointer is initially set to the value contained in the SIO call parameters DBUFLO and DBUFHI, and is then incremented by the interrupt service routines as a part of normal bus data transfer. For a command frame transfer, the pointer is set to point to the SIO-maintained command frame output buffer.

H17 BFENLO* [0034,1] and BFENHI* [0035,1] -- Buffer end address

BFENLO/BFENHI form a pointer to the byte following the last frame data byte (not including the checksum) to be sent or received.

BFENLO/BFENHI is the arithmetic sum of BUFRLO/BUFRHI plus the frame size plus -1.

Command Frame Output Buffer

See Section 9 for the command frame format and description.

H18 CDEVIC* [023A,1] -- Command frame device ID

CDEVIC is set to the value obtained by adding SIO call parameter DDEVIC to DUNIT and subtracting 1.

H19 CCOMND* [023B,1] -- Command frame command.

CCOMND is set to the value obtained from SIO call parameter DCOMND.

H20 CAUX1* [023C,1] and CAUX2* [023D,1] -- Auxiliary information

CAUX1 and CAUX2 are set to the values obtained from SIO call parameters DAUX1 and DAUX2, respectively.

Receive/Transmit Data Buffering

H21 BUFRFL* [0038,1] -- Buffer full flag

BUFRFL is a flag used by the serial bus receive interrupt service routine to indicate when the main portion of a bus frame has been received -- all but the checksum byte. BUFRFL equal to zero indicates that the main portion has not been completely received, a nonzero value indicates that the main portion has been received.

H22 RECVDN* [0039,1] -- Receive frame done flag

RECVDN is a flag used by SIO to communicate between the Serial bus receive interrupt service routine and the main SIO code. The flag is initially set to zero by SIO, and later set nonzero by the interrupt service routine after the last byte of a bus frame has been received.

H23 TEMP* [023E,1] -- SIO 1-byte I/O data

TEMP is used to receive 1-byte responses from serial bus controllers, such as ACK, NAK, COMPLETE or ERROR.

H24 XMTDON* [003A,1] -- Transmit frame done flag

XMTDON is a flag used by SIO to communicate between the Serial bus transmit interrupt service routine and the main SIO code. The flag is initially set to zero by SIO, and later set nonzero by the interrupt service routine after the last byte of a bus frame has been transmitted.

SIO Timeout

SIO uses System Timer 1 to provide the timeout capability for various operations initiated internally. See Section 6 for a discussion of the capabilities of the System Timers. TIMFLG is the flag used to communicate between SIO and the timer initiated code pointed to by CDTMA1.

H25 TIMFLG* [0317,1] -- SIO operation timeout flag

TIMFLG is used to indicate a timeout situation for a bus operation . The flag is initially set to 1, and if it attains a value of zero (after the timeout period) before the current operation is complete, the operation will be aborted. See also D2.

H26 CDTMV1* [0218,2] -- System Timer 1 value

This 2-byte count takes on various values depending upon the operation being timed. See also P4.

H27 CDTMA1* [0226,2] -- System Timer 1 address

This vector always points to the JTIMER routine, whose only function is to set TIMFLG to zero. This vector is initialized by SIO before every use, so that System Timer 1 can be used by any process that does not use SIO within a timing function. See also P5.

H28 STACKP* [0318,1] -- Stack pointer save/restore

STACKP contains the value of the 6502 SP register at entry to SIO; this is retained to facilitate a direct error exit from an SIO subroutine.

H29 TSTAT* [0319,1] -- Temporary status

TSTAT is used to return the operation status from the WAIT routine and will contain one of the SIO status byte values as shown in Appendix B.

H30 ERRFLG* [023F,1] -- I/O error flag

ERRFLG is used for communication between the WAIT routine and the outer level SIO code. ERRFLG is normally zero, but is set to $FF when a device responds with an invalid response byte.

H31 STATUS* [0030,1] -- SIO operation status

STATUS is a zero-page variable that is used within SIO to contain the operation status that will be stored to the calling sequence parameter variable DSTATS when SIO returns to the caller.

H32 SSKCTL* [0232,1] -- SKCTL copy

SSKCTL is utilized by SIO to keep track of the content of the SKCTL [D20F] register, which is a write-only register.

J. ATARI CONTROLLERS

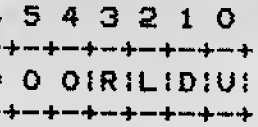
The ATARI controllers are read as part of the Stage 2 VBLANK process. The encoded data is partially decoded and processed as shown in the subsections that follow.

Joysticks

Up to four joystick controllers can be attached to the computer console, each with a 9-position joystick plus a trigger button.

J1 STICK0 - STICK3 [0278,4] -- Joystick position sense

The 4 joystick position sense variables contain a bit-encoded position sense as shown below:



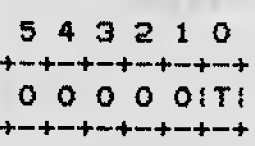
where: R = 0 indicates joystick RIGHT sensor true.
L = 0 indicates joystick LEFT sensor true.
D = 0 indicates joystick DOWN sensor true.
U = 0 indicates joystick UP sensor true.

Nine unique combinations are possible, indicating the possible joystick positions shown below:

CENTER	$0F
UP	$0E
UP/RIGHT	$06
RIGHT	$07
DOWN/RIGHT	$05
DOWN	$0D
DOWN/LEFT	$09
LEFT	$0B
UP/LEFT	$0A

J2 STRIG0 - STRIG3 [0284,4] -- Joystick trigger sense

The four joystick trigger sense variables each contain a single bit indicating the position of the joystick trigger as shown below:



where: T = 0 indicates trigger pressed.

Paddles

Up to eight paddle controllers can be connected to the computer, each with a potentiometer and a trigger sense.

J3 PADDL0 - PADDL7 [0270,8] -- Paddle position sense

There is a single-byte variable associated with each paddle position sense; the values range from 228 for full

counterclockwise rotation to 1 for full clockwise rotation.

The paddle values are often converted by the user, as shown below, to give a result of 0 for full counterclockwise rotation and 227 for full clockwise rotation:

```
VALUE := 228 - PADDLX;
```

J4 PTRIG0 - PTRIG7 [027C,8] -- Paddle trigger sense

The 8-paddle trigger sense variables each contain a single bit indicating the position of the paddle trigger as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0|T|
+-+-+-+-+-+-+-+-+
```

where: T = 0 indicates trigger pressed.

Light Pen

The OS reads the position of a single light pen and stores the horizontal and vertical position codes in two variables; these codes are not the same as the actual screen coordinates. The pen position codes for different portions of the screen are shown below:

Left edge -- 67.
Codes increase in increments of one to a value of 227, then go to 0 and continue to increase monotonically (one count per color clock).
Right edge -- 7.

Upper edge -- 16.
Codes increase in increments of one (one count per two raster lines). Lower edge -- 111.

The light pen hardware will read and latch the pen position 60 times per second, independent of the pen button position, which is separately sensed.

In order for the light pen to operate it must be positioned over a portion of the screen which has sufficient luminance to activate the photosensor in the pen; a blank (dark) screen will generally not provide enough luminance to utilize the light pen.

J5 LPENH [0234,1] -- Light pen horizontal position code

LPENH contains the horizontal position code for the light pen; the algorithm below (written in Pascal) shows the conversion from position code to screen coordinate (screen mode 7):

```
IF LPENH < 33          { check for rollover point }
   THEN                { adjust values to right of rollover }
```

```
        XPOS := LPENH + 227
      ELSE                  { no adjustment to left of rollover point }
        XPOS := LPENH;
  XPOS := XPOS - 67;    { adjust for left edge offset }
  IF XPOS < 0 THEN XPOS := 0;
  IF XPOS > 159 THEN XPOS := 159;
```

J6 LPENV [0235, 1] -- Light pen vertical position code

LPENV contains the vertical position code for the light pen; the algorithm below (written in Pascal) shows the conversion from position code to screen coordinate (screen mode 7):

```
  YPOS := LPENV - 16;   { adjust for upper edge offset }
  IF YPOS < 0 THEN YPOS := 0;
  IF YPOS > 95 THEN YPOS := 95;
```

J7 STICK0 - STICK3 [0278, 4] -- Light pen button sense

The light pen button sense is encoded in one of STICK0 - STICK3 (depending upon the actual controller port used) as shown below:

```
      7               0
     +-+-+-+-+-+-+-+-+
     |       |0|0|0|T|
     +-+-+-+-+-+-+-+-+
```

where: T = 0 indicates the light pen button is pressed.

Driving Controllers

The driving controller has no position stops and thus allows unlimited rotation in either direction; the output of the controller is a 2-bit Gray code which can be used to determine the direction of rotation. The controller is sensed using the same internal hardware as the joystick, thus the same data base variables are used for both.

J8 STICK0 - STICK3 [0278,4] -- Driving controller sense

The 4 driving controller sense variables contain an encoded rotation (position) sense value, as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|0 0 0 0 1 1|val|
+-+-+-+-+-+-+-+-+
```

where a clockwise rotation of the controller produces the following continuous sequence of four values (shown in hexadecimal):

OF, OD, OC, OE, OF, OD, ...

and a counterclockwise rotation of the controller produces the following continuous sequence of four values:

OF, OE, OC, OD, OF, OE, ...

J9 STRIG0 - STRIG3 [0284,4] -- Driving trigger sense

The four driving trigger sense variables each contain a single bit indicating the position of the driving trigger as shown below:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0|T|
+-+-+-+-+-+-+-+-+
```

where: T = 0 indicates trigger pressed.

K. DISK FILE MANAGER

See Section 5 for information relating to the Disk File Manager.

K1 FMSZPG* [0043,7] -- FMS reserved space

FMSZPG is the reserved space in the database for the variables shown below; the names associated with K2 through K5 are not in the system equate file.

K2 ZBUFP* [0043,2] -- Buffer pointer

K3 ZDRVA* [0045,2] -- Drive pointer

K4 ZSBA* [0047,2] -- Sector buffer pointer

K5 ERRNO* [0049,1] -- Error number

L. DISK UTILITY POINTER

L1 DSKUTL* [001A,2] -- Page-zero pointer variable

M. FLOATING POINT PACKAGE

See Section 8 for a description of the Floating Point Package.

M1 FR0 [00D4,6] -- FP register 0

M2 FRE* [00DA,6] -- FP register (internal)

M3 FR1 [00E0,6] -- FP register 1

M4 FR2* [00E6,6] -- FP register 2 (internal)

M5 FRX* [00EC,1] -- Spare (unused)

M6 EEXP* [00ED,1] -- Exponent value (internal)

M7 NSIGN* [00EE,1] -- Sign of mantissa (internal)

M8 ESIGN* [00EF,1] -- Sign of exponent (internal)

M9 FCHRFLG* [00F0,1] -- First character flag (internal)

M10 DIGRT* [00F1,1] -- Digits to right of decimal point

M11 CIX [00F2,1] -- Character index

M12 INBUFF [00F3,2] -- Input text buffer pointer

M13 ZTEMP1* [00F5,2] -- Temporary storage

M14 ZTEMP4* [00F7,2] -- Temporary storage

M15 ZTEMP3* [00F9,2] -- Temporary storage

M16 FLPTR [00FC,2] -- Pointer to FP number

M17 FPTR2* [00FE,2] -- FP package use

M18 LBPR1* [057E,1] -- LBUFF preamble

M19 LBPR2* [057F,1] -- LBUFF preamble

M20 LBUFF [0580,96] -- Text buffer

M21 PLYARG* [05E0,6] -- FP register (internal)

M22 FPSCR/FSCR* [05E6,6] -- FP register (internal)

M23 FPSCR1/SCR1* [05EC,6] -- FP register (internal)

M24 DEGFLG/RADFLG [00FB,1] -- Degrees/radians flag

DEGFLG = 0 indicates radians, 6 indicates degrees.

N. Power-Up and SYSTEM RESET

See Section 7 for details of the power-up and system reset operations.

RAM Sizing

During power-up and system reset the first non-RAM address above 1000 hex is located and its address retained using a nondestructive test. The first byte of every 4K memory "block" is tested to see if it is alterable; if so, the original value is restored and the next block is tested, and if not, that address is considered to be the end of RAM.

N1 RAMLO*/TRAMSZ* [0004,3] -- RAM data/test pointer (temporary)

RAMLO+1 contains the LSB of the address to be tested (always = 0) and TRAMSZ (same as RAMLO+2) contains the MSB of the address to be tested. RAMLO+0 contains the complemented value of the data originally contained in the memory location being tested.

Later in the initialization process these variables are used for totally unrelated functions; but first the value in TRAMSZ is moved to the variables RAMSIZ and MEMTOP+1.

N2 TSTDAT* [0007,1] -- Test data byte save

TSTDAT contains the original value of the memory location being tested.

Diskette/Cassette-Boot

As a part of the Power-up sequence, software can be booted from an attached disk drive or cassette player as explained in Section 10.

N3 DOSINI [000C,2] -- Diskette-boot initialization vector.

DOSINI contains the disk booted software initialization address from the beginning of the boot file (see Section 10) whenever a diskette-boot is successfully completed.

N4 CKEY* [004A,1] -- Cassette-boot request flag

CKEY is an internal flag used to indicate that the console [START] key was pressed during Power-up, thus indicating that a cassette-boot is desired. CKEY equals zero when no cassette-boot is requested, and is nonzero when a cassette-boot is requested. The flag is cleared to zero after a cassette-boot.

N5 CASSBT* [004B,1] -- Cassette-booting flag

CASSBT is used during the cassette-boot process to indicate to shared code that the cassette is being booted and not the diskette. CASSBT equal to zero indicates a diskette-boot, and nonzero indicates a cassette-boot.

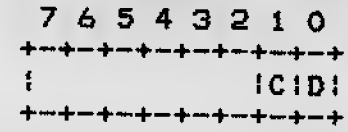
N6 CASINI [0002,2] -- Cassette-boot initialization vector

CASINI contains the cassette-booted software initialization address from the beginning of the boot file (see Section 10) whenever a

cassette-boot is successfully completed.

N7 BOOT?* [0009,1] -- Successful diskette/cassette-boot flag.

BOOT? indicates to the initialization processor which, if any, of the boot operations went to successful completion. The flag values are set by the OS and the format for the variable is shown below:



where: C = 1 indicates that the cassette-boot was completed.
D = 1 indicates that the diskette-boot was completed.

N8 DFLAGS* [0240,1] -- Diskette flags

DFLAGS contains the value of the first byte of the boot file, after a diskette-boot. See Section 10.

N9 DBSECT* [0241,1] -- Diskette-boot sector count

DBSECT is initially set to the value of the second byte of the boot file, during a diskette-boot, and is then used to control the number of additional diskette sectors read, if any.

N10 BOOTAD* [0242,2] -- Diskette-boot memory address

BOOTAD is initially set to the value of the third and fourth bytes of the boot file, during a diskette-boot, and is not modified thereafter.

Environment Control

If, at the end of a power-up or system reset, control is not given to one of the cartridges (as explained in Sections 7 and 10), then program control passes to the address contained in the data base variable DOSVEC.

N11 COLDST* [0244,1] -- Coldstart complete flag

COLDST is used by the initialization routine to detect the case of a system reset occurring before the completion of the power-up process. COLDST is set to $FF at the beginning of the power-up

sequence and is set to 0 at the completion; if a system reset occurs while the value is nonzero, the power-up sequence will be reinitiated (rather than initiating a system reset sequence).

N12 DOSVEC [000A,2] -- Noncartridge control vector

At the beginning of power-up the OS sets DOSVEC to point to the "blackboard" routine; DOSVEC can then be altered as a consequence of a diskette-boot or cassette-boot (as explained in Section 10) to establish a new control program. Control will be passed through DOSVEC on all power-up and system reset conditions in which a cartridge does not take control first.

System Reset

N13 WARMST [0008,1] -- Warmstart flag

WARMST equals $FF during a system reset (warmstart) initialization and equals 0 during a power-up initialization (coldstart).

P. INTERRUPTS

See Section 6 for a discussion of interrupt processing.

P1 CRITIC [0042,1] -- Critical code section flag

CRITIC is used to signal to the VBLANK interrupt processor that a critical code section is executing without IRQ interrupts being inhibited; the VBLANK interrupt processor will stop interrupt processing after stage 1 and before stage 2, just as if the 6502 processor I bit were set, when CRITIC is set.

CRITIC equal to zero indicates that the currently executing code section is noncritical, while any nonzero value indicates that the currently executing code section is critical.

P2 POKMSK [0010,1] -- POKEY interrupt mask

POKMSK is a software maintained interrupt mask that is used in conjunction with the enabling and disabling of the various POKEY interrupts. This mask is required because the POKEY interrupt enable register IRQEN [D20E] is a write-only register, and at any point in time the system can have several users independently enabling and disabling POKEY interrupts. POKMSK is updated by the

users to always contain the current content of IRQEN.

System Timers

The System Timers are discussed in detail in Section 6.

Realtime Clock

The realtime clock (or frame counter, as it is sometimes called) is incremented as part of the stage 1 VBLANK process as explained in Section 6.

P3 RTCLOK [0012,3] -- Realtime frame counter

RTCLOK+0 is the most significant byte, RTCLOK+1 the next most significant byte, and RTCLOK+2 the least significant byte. See the discussions at D3 and preceding B10 for OS use of RTCLOK.

System Timer 1

System Timer 1 is maintained as part of the stage 1 VBLANK process, and thus has the highest priority of any of the user timers.

P4 CDTMV1 [0218,2] -- System Timer 1 value

CDTMV1 contains zero when the timer is inactive, otherwise it contains the number of VBLANKs remaining until timeout. Also see H26.

P5 CDTMA1 [0226,2] -- System Timer 1 jump address

CDTMA1 contains the address to which to JSR should the timer timeout. See also H27 and Section 6.

System Timer 2

System Timer 2 is maintained as part of the stage 2 VBLANK process, and has the second highest priority of the user timers. The OS does not have any direct use for System Timer 2.

P6 CDTMV2 [021A,2] -- System Timer 2 value

CDTMV2 contains zero when the timer is inactive, otherwise it contains the number of VBLANKs remaining until timeout.

P7 CDTMA2 [0228,2] -- System Timer 2 jump address

CDTMA2 contains the address to which to JSR should the timer timeout. See Section 6.

System Timers 3, 4 and 5

System Timers 3, 4 and 5 are maintained as part of the stage 2 VBLANK process, and have the lowest priority of the user timers. The OS does not have any direct use for these timers.

P8 CDTMV3 [021C,2], CDTMV4 [021E,2] and CDTMV5 [0220,2]

These variables contain zero when the corresponding timers are inactive, otherwise they contain the number of VBLANKs remaining until timeout.

P9 CDTMF3 [022A,1], CDTMF4 [022C,1] and CDTMF5 [022E,2]

Each of these 1-byte variables will be set to zero should its corresponding timer timeout. The OS never modifies these bytes except to set them to zero upon timeout (and initialization).

RAM Interrupt Vectors

There are RAM vectors for many of the interrupt conditions within the system. See Section 6 for a discussion of the placing of values to these vectors.

NMI Interrupt Vectors

P10 VDSLST [0200,2] -- Display-list interrupt vector

This vector is not used by the OS. See Section 6.

P11 VVBLKI [0222,2] -- Immediate VBLANK vector

This vector is initialized to point to the OS stage 1 VBLANK

P12 VVBLKD [0224,2] -- Deferred VBLANK vector

This vector is initialized to point to the OS VBLANK exit routine. See Section 6.

IRQ Interrupt Vectors

P13 VIMIRQ [0216,2] -- General IRQ vector

This vector is initialized to point to the OS IRQ interrupt processor. See Section 6.

P14 VPRCED [0202,2] -- Serial I/O bus proceed signal

The serial bus line that produces this interrupt is not used in the current system. See Section 6.

P15 VINTER [0204,2] -- Serial I/O bus interrupt signal

The serial bus line that produces this interrupt is not used in the current system. See Section 6.

P16 V[BREAK] [0206,2] -- BRK instruction vector

This vector is initialized to point to a PLA, RTI sequence as the OS proper does not utilize the BRK instruction. See Section 6.

P17 VKEYBD [0208,2] -- Keyboard interrupt vector

This vector is initialized to point to the Keyboard Handler's interrupt service routine. See Section 6 and the discussion preceding E1.

P18 VSERIN [020A,2] -- Serial I/O bus receive data ready

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P19 VSEROR [020C,2] -- Serial I/O bus transmit ready

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P20 VSEROC [020E,2] -- Serial I/O bus transmit complete

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P21 VTIMR1 [0210,2], VTIMR2 [0212,2] and VTIMR4 [0214,2] -- POKEY timer vectors

The POKEY timer interrupts are not used by the OS See Section 6.

Hardware Register Updates

As part of the stage 2 VBLANK process, certain hardware registers are updated from OS data base variables as explained in Section 6.

P22 SDMCTL* [022F,1] -- DMA control

SDMCTL is set to a value of $02 at the beginning of a Display Handler OPEN command, and then later set to a value of $22. The value of SDMCTL is stored to DMACTL [D400] as part of the stage 2 VBLANK process.

P23 SDLSTL* [0230,1] and SDLSTH* [0231,1] -- Display list address

The Display Handler formats a new display list with every OPEN command and puts the display list address in SDLSTL and SDLSTH. The value of these bytes are stored to DLISTL [D402] and DLISTH [D403] as part of the stage 2 VBLANK process.

0360-036F	IOCB #2
0370-037F	IOCB #3
0380-038F	IOCB #4
0390-039F	IOCB #5
03A0-03AF	IOCB #6
03B0-03BF	IOCB #7

NOTE: There is a potential timing problem associated with the updating of the hardware registers from the data base variables. Since the stage 2 VBLANK process is performed with interrupts enabled, it is possible for an IRQ interrupt to occur before the updating of DLISTH and DLISTL. If the processing of that interrupt (plus other nested interrupts) exceeds the vertical-blank delay (1 msec), then the display list pointer register will not have been updated when display list processing commences for the new frame, and a screen glitch will result.

P24 GPRIOR* [026F,1] -- Priority control

The Display Handler alters bits 6 and 7 of GPRIOR as part of establishing the GTIA mode. The value of GPRIOR is stored to PRIOR [D01B] as part of the stage 2 VBLANK process.

P25 CHACT* [02F3,1] -- Character control

The Display Handler sets CHACT to $02 on every OPEN command. The value of CHACT is stored to CHACTL [D401] as part of the stage 2 VBLANK process.

P26 CHBAS [02F4,1] -- Character address base

The Display Handler sets CHBAS to $E0 on every OPEN command. The value of CHBAS is stored to CHBASE [D409] as part of the stage 2 VBLANK process. This variable controls the character subset for screen modes 1 and 2; a value of $E0 provides the capital letters and number set whereas a value of $E2 provides the lowercase letters and special graphics set. See B55 for more information.

P27 PCOLRx [02C0,4] and COLORx [02C4,5] -- Color registers

See B7 and B8.

Internal Working Variables

P28 INTEMP* [022D,1] -- Temporary storage

INTEMP is used by the SETVBL (SETVBV) routine.

R. USER AREAS

The areas shown below are available to the user in a non-nested environment. See Section 4 for further information.

R1 [0080,128]

R2 [0480,640]

ALPHABETICAL LIST OF DATA BASE VARIABLES

NAME	VID	ADDRESS SIZE
ADDCOR	D4	030E,1
ADRESS	B39	0064,2
APPMHI	A3	000E,2
ATACHR	B54	02FB,1
ATRACT	B10	004D,1
BFENHI	H17	0035,1
BFENLO	H17	0034,1
BITMSK	B50	006E,1
BLIM	D10	028A,1
BOOT?	N7	0009,1
BOOTAD	N10	0242,2
BOTSCR	B16	02BF,1
BPTR	D11	003D,1
BRKKEY	E5	0011,1
BUFADR	C1	0015,2
BUFCNT	B42	006B,1
BUFRFL	H21	0038,1
BUFRHI	H16	0033,1
BUFRLO	H16	0032,1
BUFSTR	B43	006C,2
CASBUF	D9	03FD,131
CASFLG	D8	030F,1
CASINI	N6	0002,2
CASSBT	N5	004B,1
CAUX1	H20	023C,1
CAUX2	H20	023D,1
CBAUDH	D1	02EF,1
CBAUDL	D1	02EE,1
CCOMND	H19	023B,1
CDEVIC	H18	023A,1
CDTMA1	P5,H27	0226,2
CDTMA2	P7	0228,2
CDTMF3	P9	022A,1
CDTMF4	P9	022C,1
CDTMF5	P9	022E,1
CDTMV1	P4,H26	0226,2
CDTMV2	P6	021A,2
CDTMV3	P8	021C,2

CDTMV4	P8	021E, 2
CDTMV5	P8	0220, 2
CH	E3	02FC, 1
CHKSNT	H14	003B, 1
CH1	E1	02F2, 1
CHACT	P25	02F3, 1
CHAR	B55	02FA, 1
CHBAS	P26	02F4, 1
CHKSNT	H14	003B, 1
CHKSUM	H13	0031, 1
CIOCHR	G25	002F, 1
CIX	M11	00F2, 1
CKEY	N4	004A, 1
COLAC	B24	0072, 2
COLCRS	B2	0055, 2
COLDST	N11	0244, 1
COLINC	B21	007A, 1
COLOR0	B8, P27	02C4, 1
COLOR1	B8, P27	02C5, 1
COLOR2	B8, P27	02C6, 1
COLOR3	B8, P27	02C7, 1
COLOR4	B8, P27	02C8, 1
COLRSH	B11	004F, 1
COUNTR	B23	007E, 2
CRETRY	H11	0036, 1
CRITIC	P1	0042, 1
CRSINH	B1	02F0, 1
CSTAT	S2	0288, 1
DAUX1	H9	030A, 1
DAUX2	H9	030B, 2
DBSECT	N9	0241, 1
DBUFHI	H6	0304, 1
DBUFLO	H6	0305, 1
DBYTHI	H8	0308, 1
DBYTLO	H8	0309, 1
DCB	H1	0300, 12
DCOMND	H4	0302, 1
DDEVIC	H2	0300, 1
DEGFLG	M24	00FB, 1
DELTAC	B22	0077, 2
DELTAR	B22	0076, 1
DFLAGS	N8	0240, 1
DIGRT	M10	00F1, 1
DINDEX	B35	0057, 1
DMASK	B28	02A0, 1
DOSINI	N3	000C, 2
DOSVEC	N12	000A, 2
DRETRY	H12	0037, 1
DRKMSK	B12	004E, 1
DSKTIM	C2	0246, 1
DSKUTL	L1	001A, 2
DSPFLG	B27	02FE, 1
DSTAT	B34	004C, 1

DSTATS	H5	0303,1
DTIMLO	H7	0306,1
DUNIT	H3	0301,1
DUNUSE	S3	0307,1
DVSTAT	G11	02EA,4
EEXP	M6	00ED,1
ENDPT	B25	0074,2
ERRFLG	H30	023F,1
(ERRNO	K5)	0049,1
ESCFLG	B26	02A2,1
ESIGN	M8	00EF,1
FCHRFL	M9	00F0,1
FEOF	D12	003F,1
FILDAT	B17	02FD,1
FILFLG	B18	02B7,1
FLPTR	M16	00FC,2
FMSZPG	K1	0043,7
FPSCR	M22	05E6,6
FPSCR1	M23	05EC,6
FPTR2	M17	00FE,2
FR0	M1	00D4,6
FR1	M3	00E0,6
FR2	M4	00E6,6
FRE	M2	00DA,6
FREQ	D15	0040,1
FRMADR	B41	0068,2
FRX	M5	00EC,1
FSCR	M22	05E6,6
FSCR1	M23	05EC,6
FTYPE	D13	003E,1
GPRIOR	P24	026F,1
HATABS	G12	031A,38
HOLD1	B30	0051,1
HOLD2	B31	029F,1
HOLD3	B32	029D,1
HOLD4	B20	02BC,1
HOLDCH	E7	007C,1
ICAX1	G9	034A,1
ICAX1Z	G21	002A,1
ICAX2	G9	034B,1
ICAX2Z	G21	002B,1
ICBAH	G6	0345,1
ICBAHZ	G18	0025,1
ICBAL	G6	0344,1
ICBALZ	G18	0024,1
ICBLH	G8	0349,1
ICBLHZ	G20	0029,1
ICBLL	G8	0348,1
ICBLLZ	G20	0028,1

ICCOM	G4	0342,1
ICCOMT	G23	0017,1
ICCOMZ	G16	0022,1
ICDNO	G3	0341,1
ICDNOZ	G15	0021,1
ICHID	G2	0340,1
ICHIDZ	G14	0020,1
ICIDNO	G24,G2	2002E,1
ICPTH	G7	0347,1
ICPTHZ	G19	0027,1
ICPTL	G7	0346,1
ICPTLZ	G19	0026,1
ICSPR	G10	034C,4
ICSPRZ	G22	002C,4
ICSTA	G5	0343,1
ICSTAZ	G17	0023,1
INBUFF	M12	00F3,2
INSDAT	B45	007D,1
INTEMP	P28	022D,1
INVFLG	E9	02B6,1
IOCB	G1	0340,16
IOCBAS	G13	0020,16
KEYDEL	E2	02F1,1
LBFEND	M20	0580,96
LBPR1	M18	057E,1
LBPR2	M19	057F,1
LBUFF	M20	0580,96
LINBUF	B51	0247,40
LMARGN	B5	0052,1
LOGCOL	B15	0063,1
LOGMAP	B14	02B2,4
MEMLO	A1	02E7,2
MEMTOP	A2	02E5,2
MLTTMP	B40	0066,2
NEWCOL	B19	0061,2
NEWROW	B19	0060,1
NOCKSM	H15	003C,1
NSIGN	M7	00EE,1
OLDADR	B38	005E,2
OLDCHR	B37	005D,1
OLDCOL	B3	005B,2
OLDROW	B3	005A,1
OPNTMP	B40	0066,2
PADDL0	J3	0270,1
PADDL1	J3	0271,1
PADDL2	J3	0272,1
PADDL3	J3	0273,1
PADDL4	J3	0274,1

PADDL5	J3	0275,1
PADDL6	J3	0276,1
PADDL7	J3	0277,1
PBPNT	F3	001D,1
PBUFSZ	F2	001E,1
PCOLR0	B7,P27	02C0,1
PCOLR1	B7,P27	02C1,1
PCOLR2	B7,P27	02C2,1
PCOLR3	B7,P27	02C3,1
PLYARG	M21	05E0,6
POKMSK	P2	0010,1
PRNBUF	F1	03C0,40
PTEMP	F4	001F,1
PTIMOT	F5	001C,1
PTRIG0	J4	027C,1
PTRIG1	J4	027D,1
PTRIG2	J4	027E,1
PTRIG3	J4	027F,1
PTRIG4	J4	0280,1
PTRIG5	J4	0281,1
PTRIG6	J4	0282,1
PTRIG7	J4	0283,1
RADFLG	M24	00FB,1
RAMLO	N1	0004,3
RAMSIZ	A5	02E4,1
RAMTOP	A4	006A,1
RECVDN	H22	0039,1
RMARGN	B6	0053,1
ROWAC	B24	0070,2
ROWCRS	B2	0054,1
ROWINC	B21	0079,1
RTCLOK	P3	0012,3
SAVADR	B41	0068,2
SAVIO	D7	0316,1
SAVMSC	B36	0058,2
SCRFLG	B9	02BB,1
SDLSTH	P23	0231,1
SDLSTL	P23	0230,1
SDMCTL	P22	022F,1
SHFAMT	B29	006F,1
SHFLOK	E6	02BE,1
SOUNDR	H10	0041,1
SRTIMR	E8	022B,1
SSFLAG	E4	02FF,1
SSKCTL	H32	0232,1
STACKP	H28	0318,1
STATUS	H31	0030,1
STICK0	J1,J7,J8	0278,1
STICK1	J1,J7,J8	0279,1
STICK2	J1,J7,J8	027A,1
STICK3	J1,J7,J8	027B,1
STRIG0	J2,J7,J9	0284,1

STRIG1	J2, J7, J9	0285, 1
STRIG2	J2, J7, J9	0286, 1
STRIG3	J2, J7, J9	0284, 4
SUBTMP	B48	029E, 1
SWPFLG	B44	007B, 1
TABMAP	B13	02A3, 15
TEMP	H23	023E, 1
TEMP1	D5	0312, 2
TEMP3	D6	0315, 1
TIMER1	D3	030C, 2
TIMER2	D3	0310, 2
TIMFLG	D2, H25	0317, 1
TINDEX	B49	0293, 1
TMPCHR	B33	0050, 1
TMPCOL	B46	02B9, 2
TMPLBT	B47	02A1, 1
TMPROW	B46	02B8, 1
TOADR	B40	0066, 2
TRAMSZ	N1	0004, 3
TSTAT	H29	0319, 1
TSTDAT	N2	0007, 1
TXTCOL	B4	0291, 2
TXTMSC	B52	0294, 2
TXTOLD	B53	0296, 6
TXTROW	B4	0290, 1
USAREA	R1	0080, 128
VBREAK	P16	0206, 2
VDSLST	P10	0200, 2
VIMIRQ	P13	0216, 2
VINTER	P15	0204, 2
VKEYBD	P17	0208, 2
VPRCED	P14	0202, 2
VSERIN	P18	020A, 2
VSEROC	P20	020E, 2
VSEROR	P19	020C, 2
VTIMR1	P21	0210, 2
VTIMR2	P21	0212, 2
VTIMR4	P21	0214, 2
VVBLKD	P12	0224, 2
VVBLKI	P11	0222, 2
WARMST	N13	0008, 1
WMODE	D14	0289, 1
XMTDON	H24	003A, 1
(ZBUFF	K2)	0043, 2
(ZDRVA	K3)	0045, 2
ZIOCB	G13	0020, 16
(ZSBA	K4)	0047, 2
ZTEMP1	M13	00F5, 2

```
ZTEMP3          M15         00F9,2
ZTEMP4          M14         00F7,2
```

MEMORY ADDRESS ORDERED LIST OF DATABASE VARIABLES

ADDRESS	VID	NAME
0000-0001	S7	LNZBS
0002-0003	N6	CASINI
0004-0006	N1	RAMLO, TRAMSZ
0007	N2	TSTDAT
0008	N13	WARMST
0009	N7	BOOT?
000A-000B	N12	DOSVEC
000C-000D	N3	DOSINI
000E-000F	A3	APPMHI
0010	P2	POKMSK
0011	E5	BRKKEY
0012-0014	P3	RTCLOK
0015-0016	C1	BUFADR
0017	G23	ICCOMT
001A-001B	L1	DSKUTL
001C	F5	PTIMOT
001D	F3	PBPNT
001E	F2	PBUFSZ
001F	F4	PTEMP
0020	G13, G14	ICHIDZ
0021	G15	ICDNOZ
0022	G16	ICCOMZ
0023	G17	ICOBAS
0024-0025	G18	ICBALZ, ICBAHZ
0026-0027	G19	ICPTLZ, ICPTHZ
0028-0029	G20	ICBLLZ, ICBLHZ
002A-002B	G21	ICAX1Z, ICAX2Z
002C-002F	G22, G24, G25	ICSPRZ
0030	H31	STATUS
0031	H13	CHKSUM
0032-0033	H16	BUFRLO, BUFFRHI
0034-0035	H17	BFENLO, BFENHI
0036	H11	CRETRY
0037	H12	DRETRY
0038	H21	BUFRFL
0039	H22	RECVDN
003A	H24	XMTDON
003B	H14	CHKSNT
003C	H15	NOCKSM
003D	D11	BPTR
003E	D13	FTYPE
003F	D12	FEOF
0040	D15	FREQ
0041	H10	SOUNDR
0042	P1	CRITIC
0043-0049	K1, K2, K3, K4, K5	ZBUFF, ZBUFP, ZDRVA, ZSBA
004A	N4	CKEY
004B	N5	CASSBT
004C	B34	DSTAT

004D	B10	ATRACT
004E	B12	DRKMSK
004F	B11	COLRSH
0050	B33	TMPCHR
0051	B30	HOLD1
0052	B5	LMARGN
0053	B6	RMARGN
0054-0056	B2	ROWCRS, COLCRS
0057	B35	DINDEX
0058-0059	B36	SAVMSC
005A-005C	B3	OLDROW, OLDCOL
005D	B37	OLDCHR
005E-005F	B38	OLDADR
0060-0062	B19	NEWROW, NEWCOL
0063	B15	LOGCOL
0064-0065	B39	ADRESS
0066-0067	B40	MLTTMP, OPNTMP, TOADR
0068-0069	B41	SAVADR/FRMADR
006A	A4	RAMTOP
006B	B42	BUFCNT
006C-006D	B43	BUFSTR
006E	B50	BITMSK
006F	B29	SHFAMT
0070-0073	B24	ROWAC, COLAC
0074-0075	B25	ENDPT
0076-0078	B22	DELTAR, DELTAC
0079-007A	B21	ROWINC, COLINC
007B	B44	SWPFLG
007C	E7	HOLDCH
007D	B45	INSDAT
007E-007F	B23	COUNTR
0080-00FF	SEE FLOATING POINT VARIABLE LIST AT END.	
0100-01FF	6502 STACK	
0200-0201	P10	VDSLST
0202-0203	P14	VPRCED
0204-0205	P15	VINTER
0206-0207	P16	VBREAK
0208-0209	P17	VKEYBD
020A-020B	P18	VSERIN
020C-020D	P19	VSEROR
020E-020F	P20	VSEROC
0210-0215	P21	VITMR1, VITMR2, VITMR4
0216-0217	P13	VIMIRQ
0218-0219	P4, H26	CDTMV1
021A-021B	P6	CDTMV2
021C-0221	P8	CDTMV3, CDTMV4, CDTMV5
0222-0223	P11	VVBLKI
0224-0225	P12	VVBLKD
0226-0227	P5, H27	CDTMA1
0228-0229	P7	CDTMA2
022A	P9	CDTMF3

022B	E8	SRTIMR
022C	P9	CDTMF4
022D	P28	INTEMP
022E	P9	CDTMF5
022F	P22	SDMCTL
0230-0231	P23	SDLSTL, SDLSTH
0232	H32	SSKCTL
023A	H18	CDEVIC
023B	H19	CCOMND
023C-023D	H20	CAUX1, CAUX2
023E	H23	TEMP
023F	H30	ERRFLG
0240	N8	DFLAGS
0241	N9	DBSECT
0242-0243	N10	BOOTAD
0244	N11	COLDST
0246	C2	DSKTIM
0247-026E	B51	LINBUF
026F	P24	GPRIOR
0270-0277	J3	PADDL0 -- PADDL7
0278-027B	J1, J7, J8	STICK0 -- STICK3
027C-0283	J4	PTRIG0 -- PTRIG7
0284-0287	J2, J7, J9	STRIG0 -- STRIG3
0289	D14	WMODE
028A	D10	BLIM
028B-028F	S10	unused
0290-0292	B4	TXTROW, TXTCOL
0293	B49	TINDEX
0294-0295	B52	TXTMSC
0296-029B	B53	TXTOLD
029D	B32	HOLD3
029E	B48	SUBTMP
029F	B31	HOLD2
02A0	B28	DMASK
02A1	B47	TMPLBT
02A2	B26	ESCFLG
02A3-02B1	B13	TABMAP
02B2-02B5	B14	LOGMAP
02B6	E9	INVFLG
02B7	B18	FILFLG
02B8-02BA	B46	TMPROW, TMPCOL
02BB	B9	SCRFLG
02BC	B20	HOLD4
02BE	E6	SHFLOK
02BF	B16	BOTSCR
02C0-02C3	B7, P27	PCOLR0 -- PCOLR3
02C4-02C8	B8, P27	PCOLR0 -- PCOLR4
02E4	A5	RAMSIZ
02E5-02E6	A2	MEMTOP
02E7-02E8	A1	MEMLO
02EA-02ED	G11	DVSTAT
02EE-02EF	D1	CHBAUDL, CHBAUDH
02F0	B1	CRSINH
02F1	E2	KEYDEL

02F2	E1	CH1
02F3	P25	CHACT
02F4	P26	CHBAS
02FA	B55	CHAR
02FB	B54	ATACHR
02FC	E3	CH
02FD	B17	FILDAT
02FE	B27	DSPFLG
02FF	E4	SSFLAG
0300	H1, H2	DCB/DDEVIC
0301	H3	DUNIT
0302	H4	DCOMND
0303	H5	DSTATS
0304-0305	H6	DBUFLO, DBUFHI
0306	H7	DTIMLO
0308-0309	H8	DBYTLO, DBYTHI
030A-030B	H9	DAUX1, DAUX2
030C-030D	D3	TIMER1
030E	D4	ADDCOR
030F	D8	CASFLG
0310-0311	D3	TIMER2
0312-0313	D5	TEMP1
0315	D6	TEMP3
0316	D7	SAVIO
0317	D2, H25	TIMFLG
0318	H28	STACKP
0319	H29	TSTAT
031A-033F	G12	HATABS
0340	G1, G2	IOCB, ICHID
0341	G3	ICDNO
0342	G4	ICCOM
0343	G5	ICSTA
0344-0345	G6	ICBAL, ICBAH
0346-0347	G7	ICPTL, ICPTH
0348-0349	G8	ICBLL, ICBLH
034A-034B	G9	ICAX1, ICAX2
034C-034F	G10	ICSPR
0350-035F	G2-G10	(IOCB #1)
0360-036F	G2-G10	(IOCB #2)
0370-037F	G2-G10	(IOCB #3)
0380-038F	G2-G10	(IOCB #4)
0390-039F	G2-G10	(IOCB #5)
03A0-03AF	G2-G10	(IOCB #6)
03B0-03BF	G2-G10	(IOCB #7)
03C0-03E7	F1	PRNBUF
03FD-047F	D9	CASBUF
0480-06FF	R2	User Area

FLOATING POINT PACKAGE VARIABLES

00D4-00D9	M1	FR0
00DA-00DF	M2	FRE
00E0-00E5	M3	FR1
00E6-00EB	M4	FR2
00EC	M5	FRX
00ED	M6	EEXP
00EE	M7	NSIGN
00EF	M8	ESIGN
00F0	M9	FCHRFLG
00F1	M10	DIGRT
00F2	M11	CIX
00F3-00F4	M12	INBUFF
00F5-00F6	M13	ZTEMP1
00F7-00F8	M14	ZTEMP4
00F9-00FA	M15	ZTEMP3
00FB	M24	RADFLG/DEGFLG
00FC-00FD	M16	FLPTR
00FE-00FF	M17	FPTR2
057E	M18	LBPR1
057F	M19	LBPR2
0580-05FF	M20	LBFEND,LBUFF
05E0-05E5	M21	PLYARG
05E6-05EB	M22	FPSCR/FSCR
05EC-05F1	M23	FPSCR1/SCR1

INDEX

The subject index contains three forms of references:

Section number, such as '3.'
Appendix, such as 'App B'
Variable ID from Appendix L, such as 'B7'.

ATARI standards 12
ATASCII B54-55, 5, App D-G
attract mode B10-12, 6,

bit mapped graphics B28-B29, 5, App H
blackboard mode 3, N12, 7, 12
BNF 1
boot 3, 4, N3-10, 5, 7, 10
BREAK E5, 6, 12

cartridge 3, 4, 7, 10
cassette baud rate determine D1-D7
cassette-boot 3, N3-10, 7, 10
cassette device D1-D15, 3, 5
Cassette Handler (C) 5
CIO (Central I/O Utility) G1-25, 5, 9
CIO/user interface G1-11, 5, App A, App B
CIO/Handler interface G12-22, 9
CLOSE I/O command 5, 9
coldstart (see 'Power-up')
color control B7-8, 5, 6
control characters B26-27, 5, App D
critical section P1, 6
cursor B1-4, 5

database 4
DCB (Device Control Block) H1-9, 5, 9
DELETE I/O command 5
development system 13
device/filename specification 5
Device Handler 5, 9
device table 2, G12, 5, 7, 9
disk-boot 3, N3-10, 5, 7, 10
disk device 5
Disk File Manager (D) K1-5, 5
Disk Handler (resident) C1-2, 5
display device (screen) B54-55, 5, App E, App H
Display Handler (S) B1-55, 5
display list 4, P10
DOS (Disk Utilities) L1, 12
DRAW I/O command B17-25, 5
driving controller J8-9
Educational System Format Cassettes 5
error handling G5, H5, H11-12, 9, App B-C

EOF (end-of-file) 5

File Management System 5
FILL I/O command B17-25, 5
floating point package 2, 4, M1-24, 8, App J
FORMAT I/O command 5
free memory 4, A1-3, R1-2, 4, 7

game controllers 3, J1-9, 6, 11
GET CHARACTER I/O command 5, 9
GET RECORD I/O command 5, 9
GET STATUS I/O command G11, 5, 9

Handler (see 'device handler' and individual device handlers)

initialization, cartridge 7
initialization, Handler 7, 9
initialization, interrupt 6
initialization, system 4, 7, 10
internal display code 5, B54
interrupts 2, P1-28, 6
interrupt mask P2, 6
inverse video (display) E9, 5
I/O 2, 4, 5, 9
IOCB (I/O Control Block) G1-10, 5, 9
I/O retry logic H11-12

joystick J1-2

keyboard Autorepeat E8
keyboard device 5
Keyboard Handler (K) E1-9, 5, App F
keyboard key debouncing E1-3

light pen 11, App J
LNBUG 13
LOCK I/O command 5
logical text lines (screen) B14-15, 5

memory (see 'RAM', 'ROM' and 'free memory')
memory dynamics A1-5, N1-2, 4, 5, 7
memory map 4

NOTE I/O command 5

OPEN I/O command 5, 9

paddle J3-4
page 0 4, M1-17, R1, 9
page 1 4, 9
peripheral devices 3
POINT I/O command 5
Power-up 2, N1-13, 4, 7, 12
printer device 5, App G

Printer Handler (P) F1-5, 5
program development 13
PUT CHARACTER I/O command 5, 9
PUT RECORD I/O command 5, 9

RAM 3, 4, 9
record (I/O) 5
RENAME I/O command 5
RESET 2, N1-13, 6, 7, 12
ROM (OS) 1, 4
RS-232-C Handler (R) 5, 9

Screen Editor (E) B1-55, 5
screen margins B5-6, 5, 7
screen modes 4, 5, App H
scrolling (text) B9, 5
serial I/O bus 3, 5, 9, App I
[SHIFT]/CONTROL lock E6-7, 5
SIO (Serial bus I/O Utility) H1-32, P13-21, 5, 9, App C
sound control (SIO) H10, 11
SPECIAL I/O commands 5, 9
split screen B16, 5
stack 4
start/stop (display) E4, 5, 12
stage 1 VBLANK process P3-5, 6
stage 2 VBLANK process P6-9, P22-27, 6

tabs (Screen Editor) B13, 5
timeout (device) H25-27, 9
timers (system) P3-9, 6

UNLOCK I/O command 5
user workspace 4, M18-23, R2

vectors, RAM P5, P7, P10-21, 6, 9
vectors, ROM 5, 9, App J
vertical blank interrupt P11-12, 6

warmstart (see 'RESET')
wild-card (disk filename) 5

ZIOCB (Zero-page IOCB) G13-22, 9, 0020,16